

# PowerPC 823 Diab Data Tools

## Nucleus Target Specific Notes

0001019-001 rev.100



Copyright (c) 1999  
Accelerated Technology, Inc.  
720 Oak Circle Dr. E.  
Mobile, AL 36609  
(334) 661-5770



## Related Documentation

**Nucleus PLUS Reference Manual**, by Accelerated Technology, describes the operation and usage of the Nucleus PLUS kernel.

**Nucleus PLUS Internals**, by Accelerated Technology, describes, in considerable detail, the implementation of the Nucleus PLUS kernel.

**Nucleus NET Reference Manual**, by Accelerated Technology, describes the operation and usage of Nucleus NET.

**Nucleus TFTP Server Reference Manual**, by Accelerated Technology, describes the operation and usage of Nucleus TFTP Server.

**Nucleus TFTP Client Reference Manual**, by Accelerated Technology, describes the operation and usage of Nucleus TFTP Client.

**Nucleus SNMP Reference Manual**, by Accelerated Technology, describes the operation and usage of Nucleus SNMP.

**Nucleus C++ Reference Manual**, by Accelerated Technology, describes the operation and usage of Nucleus C++.

**Nucleus GRAFIX Reference Manual**, by Accelerated Technology, describes the operation and usage of Nucleus GRAFIX.



## Style and Symbol Conventions

Program listings, program examples, filenames, menu items/buttons and interactive displays are each shown in a special font.

Program listings and program examples - `Courier New`

Filenames - `COURIER NEW, ALL CAPS`

Interactive Command Lines - **`Courier New, Bold`**

Menu Items/Buttons – *Times New Roman Italic*

## Trademarks

MS-DOS is a trademark of Microsoft Corporation

UNIX is a trademark of X/Open

IBM PC is a trademark of International Business Machines, Inc.

Diab Data is a trademark of Diab, Inc.

## Additional Assistance

For additional assistance, please contact us at the following:

Accelerated Technology

720 Oak Circle Drive, East

Mobile, AL 36609

800-468-6853

334-661-5770

334-661-5788 (fax)

[support@atinucleus.com](mailto:support@atinucleus.com)

<http://www.atinucleus.com>

Copyright (©) 1999, All Rights Reserved.

Document Part Number : 0001019-001 rev.100

Last Revised: December 1, 1999



# Contents

Chapter 1 - Introduction .....	1
Installing the Nucleus Software.....	2
Nucleus Distribution .....	2
Additional Documentation .....	3
Chapter 2 – Nucleus PLUS .....	5
Nucleus PLUS Software.....	6
Tools and Hardware .....	6
Building the Nucleus PLUS Library From a DOS Prompt.....	6
Command Line Parameters for Library Building.....	7
Conditional Compilation for Library Building .....	8
Building an Application .....	8
Command Line Parameters for Application Building .....	9
Conditional Compilation for Application Building.....	9
SingleStep Setup.....	9
Building and Executing the PLUS Demonstration System	
From a DOS Prompt .....	10
FADS board.....	11
EST board.....	11
Running the Demonstration.....	12
RTA Integration.....	12
Profiling .....	13
Run-time Error Checking.....	14
Chapter 3 – Nucleus PLUS Usage .....	17
Data Types.....	18
System Startup.....	18
User Modifications.....	19
Execution Mode.....	19
Register Usage.....	19
Memory Usage .....	20
Nucleus PLUS Memory Usage .....	20
Stacks.....	20
Minimum Stack Size .....	21
Tuning Task/HISR Stack Size .....	21



## Nucleus Target Specific Notes PowerPC 823 Diab Data Tools

Interrupt Vectors .....	22
Interrupt Control .....	23
Interrupt Control Services .....	24
Initial Interrupt Lockout.....	24
Timer Interrupt.....	24
Reentrancy .....	24
<b>Chapter 4 – Nucleus PLUS Control Block Offsets .....</b>	<b>25</b>
Task Control Block (TCB) Offsets.....	26
HISR Control Block (HCB) Offsets .....	27
<b>Chapter 5 - Nucleus C++.....</b>	<b>29</b>
Nucleus C++ Software .....	30
Tools and Hardware.....	30
Building the Nucleus C++ Library From a DOS Prompt.....	30
Compiler Options .....	31
Conditional Compilation.....	31
Building and Executing the C++ Demonstration System From a DOS Prompt.....	32
System Startup .....	33
Operators New and Delete .....	33
<b>Chapter 6 – Nucleus NET .....</b>	<b>35</b>
Introduction .....	36
Tools and Hardware.....	37
Building The Nucleus NET Library From a DOS Prompt.....	37
Compilation Switches Used .....	37
Defines Used for Compilations .....	38
TARGET.H.....	39
Nucleus NET Interface Routines.....	39
Overview of the Demonstration System .....	39
Using the Nucleus NetDemo Application .....	39
Settings Common to all the NET Demos .....	41
TCP Server Demo.....	41
TCP Client Demo .....	45
UDP Client Demo.....	48
UDP Server Demo.....	50
Stopping the Test Application.....	55
Exiting the Program.....	55
<b>Chapter 7 - Nucleus MPC823 Ethernet Driver.....</b>	<b>57</b>
Introduction .....	58
Tools and Hardware.....	59
Building the Nucleus Ethernet Library From a DOS Prompt.....	59
Description of Driver Files.....	59
Building and Executing the Ethernet Demonstration System .....	60



Chapter 8 – Nucleus GRAFIX.....	63
Introduction.....	64
Tools and Hardware.....	64
Building the Nucleus GRAFIX GS Library From a DOS Prompt.....	65
Building and Executing the GRAFIX GS Demonstration System	
From a DOS Prompt.....	65
Building the Nucleus GRAFIX WT Library From a DOS Prompt.....	65
Building and Executing the GRAFIX WT Demonstration System	
From a DOS Prompt.....	65
Display Modes.....	66
GS Configuration.....	66
WT Configuration.....	67
Chapter 9 – Nucleus TFTP Server.....	71
Introduction.....	72
Tools and Hardware.....	72
Building the Nucleus TFTP Server Library From a DOS Prompt.....	73
Building and Executing the TFTP Server Demonstration System	
From a DOS Prompt.....	73
Chapter 10 – Nucleus TFTP Client.....	75
Introduction.....	76
Tools and Hardware.....	76
Building the Nucleus TFTP Client Library From a DOS Prompt.....	77
Building and Executing the TFTP Client Demonstration System	
From a DOS Prompt.....	77
Chapter 11 – Nucleus SNMP.....	79
Introduction.....	80
Tools and Hardware.....	81
Rebuilding other Nucleus Libraries.....	81
Building the Nucleus SNMP Library From a DOS Prompt.....	81
Building and Executing the SNMP Demonstration System From a DOS Prompt.....	82
Chapter 12 – Nucleus EDE.....	83
Configuration for Nucleus EDE.....	84
Demonstration System.....	84
Setting the Tool Directory.....	85
Exclude From Build.....	86
Starting a Debugger through EDE.....	87





**1**

# Introduction

**Installing the Nucleus Software**

**Nucleus Distribution**

**Additional Documentation**



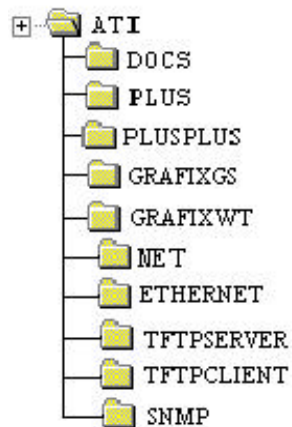
### Installing the Nucleus Software

The Nucleus distribution software for the MPC823 environment contains all the files needed for operation, as well as an example system for each of the products you have purchased. You will need to run the program "SETUP.EXE," which is located on the CD you received from Accelerated Technology, Inc. After running this program, follow the directions on your screen. This program will install the software you have purchased and place it in the directory you have provided.

### Nucleus Distribution

Nucleus PLUS and ATI's other products expect a particular directory structure to exist for the build process to operate correctly. You are free to modify this directory structure, however, some modification will be necessary for a clean operation of the associated batch files and/or project files.

The directory structure written by ATI's installation program will be as follows:



This is only a sample listing of ATI's products. The actual directory structure may be different, depending on the products you have purchased.

## Additional Documentation

Additional documentation for any products that you have purchased will be located in the \DOCS directory where the Nucleus software was installed. As an example, this directory defaults to C:\ATI. Therefore, additional documentation would be located in the C:\ATI\DOCS directory





# Nucleus PLUS

## Nucleus PLUS Software

Building the Nucleus PLUS Library  
From a DOS Prompt

Building an Application

Building and Executing the PLUS  
Demonstration System From a  
DOS Prompt

RTA Integration

Profiling

Run-time Error Checking



### Nucleus PLUS Software

The Nucleus PLUS distribution software for the PowerPC 823 environment contains all the files needed for operation, as well as an example system. Information in this chapter deals with specific requirements of the PowerPC 823 using Diab Data tools. Examples in the following sections are given for Diab Data tools hosted on a Windows system.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

Nucleus PLUS is implemented as a C library. To install the Nucleus PLUS software, simply execute the `SETUP.EXE` from the distribution software. The Nucleus PLUS library and demo rely on this structure and will not build correctly if this directory structure is modified. After installation, the directory structure should resemble the following:



### Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.3B
Assembler:	DAS V4.3B
Linker:	DLD V4.3B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola FADS and EST*

(\*programs for the EST are compile only)

### Building the Nucleus PLUS Library From a DOS Prompt

In order to build the demonstration system, you must first build the library.

The MS-DOS batch file `PLUS.BAT` contains the Diab Data assembler, compiler, and archiver commands necessary to build the Nucleus PLUS archive.

The Nucleus PLUS library builds for the MPC823 FADS board by default. The library may be built for the EST by changing the target value at the top of `PLUS.BAT` or use the command-line parameter `EST` when executing `PLUS.BAT`.



PLUS.BAT creates the Nucleus PLUS library (\PLUS\O\PLUS.LIB). The default compiles for the MPC823 FADS board. If you have one of the other 823 boards, use one of the following as a command-line parameter:

FADS - Motorola 823 FADS board

EST - EST 823 Pro board

The Nucleus library can be built by executing the following command :

```
\ATI\PLUS> plus fads <cr> -- Builds for the Motorola FADS board.
```

or

```
\ATI\PLUS> plus est <cr> -- Builds for the EST 823 Pro board.
```

The previous commands assume the existence of two files, namely DEMO.LNK and DEMO.C.

The file CRT0.S is supplied by the Diab Data tools and provides some run-time initialization. INT\_FADS.S OR INT\_EST.S contains all of the low-level Nucleus initialization and the hardware initialization for the evaluation board. USER\_MMU.C and MMU.S contain the initialization of the memory management unit and the setup of user-defined memory regions in the translation lookaside buffers. UART.C provides a driver for the SMC1 channel on the MPC823.

### Command Line Parameters for Library Building

There are several command line parameters associated with compilation / assembly / archiving of each library element. The following is a description of each command line option for the Diab Data development tools on an MS-DOS host:

Command Line Option	Meaning
-c	Specifies an ".O" file, bypassing the link phase of the compiler.
-g	Generates symbolic debugger information, with most optimization turned off.
-r	Replaces the named ".O" files in the archive "PLUS.LIB".
-sr	Updates the symbol table in the archive file and sorts the object files in the archive.
-tPPC823ES	Compiler generates an ELF format object file and uses software floating-point.
-DFADS	Used in USER_MMU.C for FADS board.
-DEST	Used in USER_MMU.C for EST board.



### Conditional Compilation for Library Building

The Nucleus PLUS library has several conditional compilation options. These conditional compilation flags enable various features within the Nucleus PLUS library source.

The conditional compilation options and their specification with the Diab Data development tools is as follows:

Compilation Flag	Meaning
-DNU_ENABLE_HISTORY	Enables history saving in the specified file. Note: only files of the form <code>**C.C</code> are affected by this option.
-DNU_ENABLE_STACK_CHECK	Enables stack checking at the beginning of each function in the specified file. Note: only files of the form <code>**C.C</code> are affected by this option.
-DNU_ERROR_STRING	Enables making an ASCII error string if a fatal system error occurs. This flag is applicable to the compilation of <code>ERD.C</code> , <code>ERI.C</code> , and <code>ERC.C</code> .
-DNU_NO_ERROR_CHECKING	Disables error-checking shell on creation of the timer HISR in <code>TMI.C</code> . Not applicable to any other Nucleus PLUS library compilation.
-DNU_DEBUG	Maps the data structures used by the application and defined in <code>NUCLEUS.H</code> to the actual internal data structures in Nucleus PLUS. If this option is used, it is typically a good idea to compile the entire library with it.
-DNU_INLINE	Replaces some linked-list processing with in-line code in order to improve performance. This option is applicable to any or all <code>**C.C</code> or <code>**S.C</code> files.

### Building an Application

Building an application is described in the “Getting Started” chapter of the *Nucleus PLUS Reference* manual. Assuming that `NUCLEUS.H`, `PLUS.LIB` and `DEMO.LIB` are accessible from the current directory, the following Diab Data commands (in **bold** type) would build and link a simple user application:

```
> dcc -v -g -tPPC823ES -c demo.c
> dld -lc -m2 -o demo.elf demo.o plus.lib demo.lnk > o\demo.map
```

The `DEMO.LNK` file is used by the linker to handle all the loading of files, linking any of the routines needed from the archive, supplying starting addresses for memory, variables, the `.bss` section, etc. The file `DEMO.C` contains the function, `Application_Initialize`, as well as all tasks and other structures. The `DEMO.O` file needs to be linked together with `PLUS.LIB`.



## Command Line Parameters for Application Building

There are several command line parameters associated with compilation/assembly of application elements. The following is a description of each command line option for the Diab Data development tools on an MS-DOS host:

Command Line Option	Meaning
<b>-c</b>	Specifies an ".o" file, bypassing the link phase of the compiler.
<b>-g</b>	Generates symbolic debugger information, with most optimization turned off.
<b>-o</b>	Specifies the output file name for the linker.
<b>-m2</b>	Generates a map file.
<b>-tPPC823ES</b>	Compiler generates an ELF format object file and uses software floating-point.

## Conditional Compilation for Application Building

Nucleus PLUS application elements may disable error checking on parameters supplied to Nucleus PLUS services by supplying the **-DNU\_NO\_ERROR\_CHECKING** command line option to the compiler. This results in a substantial increase in run-time performance, and also may reduce code size.

The application data structures defined in `NUCLEUS.H` may be mapped directly to the internal Nucleus PLUS data structures by defining `NU_DEBUG` with a **-D** option during compilation. This allows the user to directly examine the internals of each Nucleus PLUS data structure within a source-level debugging environment. If the `NU_DEBUG` option is used, it is often a good idea to re-build all of the Nucleus PLUS source code with the `NU_DEBUG` option.

## SingleStep Setup

Prior to launching the SDS V7.4 debugger, SDS must be set up to properly configure the board, which you will be using. For convenience, several SDS SingleStep configuration and debugger files are included in the `\PLUS\DEMO` subdirectory. The files consists of:

```
EST823.CFG          EST823UP.DBG          FADSFZR0220.CFG
UPMFADS0220.DBG
```

These files should be copied into the `\CMD` subdirectory of the SingleStep installation. Then, in the `\INIT` subdirectory of the same installation, an `SSTEP.INI` file should be created, with the following contents:

```
alias_config'source C:\sds74\cmd\fadsfrz0220.cfg'- For FADS board
or
alias _config 'source C:\sds74\cmd\est823.cfg'      - For EST board
```



## Nucleus Target Specific Notes PowerPC 823 Diab Data Tools

These configuration files have been modified to prevent SingleStep from trapping the decrementer or external interrupts, which are required by Nucleus PLUS and the demonstration program to function correctly. Note the paths above assume your SDS installation is installed on the C drive and named SDS74. Modifications may be necessary for your environment.

## Building and Executing the PLUS Demonstration System From a DOS Prompt

The Nucleus PLUS software contains all the files necessary to build a demonstration system. The demonstration system is designed to execute on the Motorola FADS board or the EST 823 Pro board using the SingleStep debugger.

The Nucleus demonstration system can be built by executing the following command:

```
\ATI\PLUS\DEMO> demo <cr>
```

The DEMO.BAT batch file compiles the DEMO.C file. The demo batch files then links all the files together and generates the map file and the ELF file for downloading to the board. This build process should be used as an example for any user-created applications.



A file, named `DEMO.ELF`, is produced by the `DEMO.BAT` batch file. This file is ready to be downloaded by the SingleStep On-Chip (MPC8XX) BDM debugger. Once the SingleStep On-Chip (MPC8XX) BDM has been started, it will prompt you for the demo file you wish to load or you may load the file by doing the following:

- Choose *File* and then select *Debug* in the SDS debugger to upload the
- `O\DEMO.ELF` executable to the target.
- Select the *Processor* tab, and select “*By Object File*”.
- Select the *Options* tab and uncheck “*Require Exact Symbol Names*”.
- The remaining options are checked. Click *OK* to upload.

After the code has downloaded, click on the “*green light*” button to start the demo. The demo prints status information out of the serial port. Depending on what board is being used, the serial demo may be viewed by one of the following ways:

### FADS board

Connect a straight-through cable from the bottom DB9 port of the FADS board to an available COM port of your PC.

For a communication program that will be used to check output use the following settings:

baud	115200 bits per second
data bits	8
parity	none
stop bits	1
flow control	none

### EST board

Connect a null modem cable from the JP12 port of the EST board to an available COM port of your PC.

**Note:** Be sure to check that the value `INT_System_Clock` is correct in either the `INT_FADS.S` or `INT_EST.S`. Depending on the crystal used, the eval board runs at 40 MHz or 50 MHz. If the `INT_System_Clock` variable does not match the true processor clock, the serial output will be garbled.



### Running the Demonstration

After the demo program has run for a few minutes, there are some variables that can be examined to ensure the program is running properly. These variables should be put into the Watch Window of the debugger. The variables are summarized in the following table:

Variable	Function	Description
Task_Time	task_0	Number of times task_0 has been to sleep and awakened.
Task_1_messages_sent	task_1	Number of messages task_1 has sent to task_2.
Task_2_messages_received	task_2	Number of messages task_2 has received from task_1.
Task_2_invalid_messages	task_2	Number of unsynchronized messages task_2 has received from task_1.
Event_Detections	task_5	Number of times task_5 has received an event from task_0.

Task\_Time and Event\_Detections should be equal, or differ by no more than one.  
Task\_2\_messages\_received should be within 100 messages of Task\_1\_messages\_sent.  
Task\_2\_invalid\_messages should be 0.

### RTA Integration

This port supports the profiling and run-time error checking features of the Diab Data RTA suite. In order to use these features, you must have the following:

- Diab Data 4.3b or higher
- SingleStep 7.4 or higher
- You must have installed the RTA component of SingleStep
- If you are using Nucleus EDE, it must be 3.1 or higher

By default, Nucleus PLUS will not have the RTA features enabled when it is compiled as described on the preceding pages. The RTA features are enabled by compiling the PLUS library and application with additional flags. Batch files are provided which compile the PLUS library and application with the added flags along with the default flags. Nucleus PLUS should be enabled with only one of the RTA features at a time, NEVER both of them at the same time.



## Profiling

To instrument all of the generic Nucleus PLUS code for profiling, use `PLUSPROF.BAT` in place of `PLUS.BAT`. `PLUSPROF.BAT` is used exactly like `PLUS.BAT` as described in the section on building the Nucleus PLUS library. To build the demonstration program for profiling, use `DEMOPROF.BAT` in place of `DEMO.BAT`. When building any application, `DEMOPROF.BAT` should be used as a reference. Any C files that you want profiled must be built with the `-Xprof-all` and `-DNU_RTA_PROF` flags in addition to the default flags.

The application modules must be linked with the RTA library in addition to the PLUS library, by adding the `-lrtc` flag to the linker command.

There are two utilities required by RTA for it to gather profiling data from the target and display the data in graphical form. These programs are provided in the `\PLUS` directory of the Nucleus PLUS distribution.

`RTASSTEP.EXE` should be copied to the `\diab\4.3b\win32\targcomm\sstep` directory. `PROFILER_COLLECT_DATA.DBG` should be copied to the `\cmd` directory of the SingleStep installation. These utilities may be modified and updated by Diab Data, in which case you should use the Diab versions instead of the ones provided in the Nucleus PLUS distribution.

If you are using Windows NT, you must make the `LM_LICENSE_FILE` environment variable that Diab Data uses available to the NT system. From the 'START' menu, choose *SETTINGS/CONTROL PANEL*, then double-click the 'SYSTEM' icon. In the resulting dialogue box, choose the 'ENVIRONMENT' tab, and add the `LM_LICENSE_FILE` environment variable, setting it to point to the Diab Data license file.

If you are using Windows 95/98, make sure `LM_LICENSE_FILE` is in `AUTOEXEC.BAT`. The demonstration program `DEMO.ELF` should be downloaded to the target board as described in the preceding pages. After the program has been downloaded and the DEBUG dialogue box has been cleared from the screen, the RTA component of SingleStep must be configured. In the button bar, SingleStep provides three RTA-related buttons. If these buttons do not appear, choose RTA PROFILER in the 'TOOLBARS' menu to enable them.

Click on the first RTA button to bring up the RTA configuration dialogue box. Assuming you are using the default installation directories, enter the following values:

Profiler Path:	<code>c:\diab\4.3b\WIN32\bin\rtc.exe</code>
Command Parameters:	<code>-cd c:\ati\plus\demo\o</code> <code>c:\ati\plus\demo\o\rtc_db</code>
Profiler data file:	<code>c:\ati\plus\demo\o\demo.prf</code>

Although these values are for the Nucleus PLUS demo, they should be referred to as an example for your own application.



## Nucleus Target Specific Notes PowerPC 823 Diab Data Tools

After the RTA has been configured, the program you downloaded can be started. While it runs, profiling data will be saved on the target. After stopping execution, bring up the SingleStep console window, then click on the third RTA button. The debugger will print a message to the console, upload the profiling data from the target, then print a message indicating that the data has been uploaded.

Click on the second RTA button to start the RTA graphical program display. If the RTA program is running for the first time, it will ask to create a directory to store user information. If RTA doesn't find a previously created `\rta_db` subdirectory, it will ask to create one as specified by the *Command Parameters* setting of the RTA configuration.

Choose *File/Open* from the menu and load `DEMO.ELF`. Choose *Application/Target Settings...* from the menu. In the resulting dialogue box, highlight `sds_singlestep`, then click on the OK button. You are now ready to connect to an RTA server. Choose *Application/Connect* from the menu to start the RTA server. The RTA server will print messages to the RTA output log window, the last one indicating it is ready for commands. To read and display the profiling data, choose *Application/Snapshot* from the menu. Please refer to the RTA user's manual for details about the graphical displays and other features of RTA.

## Run-time Error Checking

To instrument all of the generic Nucleus PLUS code for run-time error checking, use `PLUSRTC.BAT` in place of `PLUS.BAT`. `PLUSRTC.BAT` is used exactly like `PLUS.BAT` as described in the section on building the Nucleus PLUS library.

By default, `DEMO.C` does not contain any code that the run-time error checker will complain about. To see the run-time error checker actually flag a problem, insert the following line of code into any one of the functions of `DEMO.C`:

```
NU_Deallocate_Memory(&Task_Time);
```

This line of code attempts to free global memory, which is a violation the run-time error checker will report.

To build the demonstration program for run-time error checking, use `DEMORTC.BAT` in place of `DEMO.BAT`. When building any application, `DEMORTC.BAT` should be used as a reference. Any C files that you want checked at run-time must be built with the `-Xrtc` and `-DNU_RTA_RTC` flags in addition to the default flags. The application modules must be linked with the RTA library in addition to the PLUS library, by adding the `-lrta` flag to the linker command.



The demonstration program `DEMO.ELF` should be downloaded to the target board as described in the preceding pages. After the program has been started, bring up the SingleStep console window to see the message the RTA printed about the memory violation. All of the run-time error messages are displayed on the SingleStep console. Please refer to the RTA user's manual for more information on the types of errors the run-time error checker deals with and the messages it displays.





# 3

## Nucleus PLUS Usage

Data Types

System Startup

Execution Mode

Register Usage

Memory Usage

Stacks

Interrupt Vectors

Interrupt Control

Timer Interrupt

Reentrancy



### Data Types

Since different processor architectures and even different compilers define C data types differently, Nucleus PLUS defines a set of its own data types. These data types are mapped to the compiler data types that have the required capability.

Nucleus PLUS defines several data types in the include file `NUCLEUS.H`. The data type assignments for the Diab Data compiler are as follows:

Nucleus PLUS Type	Actual Data Type
UNSIGNED	unsigned long
SIGNED	long
OPTION	unsigned long
DATA_ELEMENT	unsigned long
STATUS	int
UNSIGNED_CHAR	unsigned char
CHAR	char
INT	int
VOID	void
UNSIGNED_PTR	unsigned long *
BYTE_PTR	unsigned char *

### System Startup

The label `start` is defined at the reset vector and marks the entry point of the system. The reset vector calls the Nucleus function `INT_Initialize`.

`INT_Initialize` is responsible for initializing the board, setting up the interrupt vector table, and setting up several system stack areas. `INT_Initialize` calls the Diab Data function `_init_main`, which copies the data section from ROM to RAM and clears the `.BSS` section. After `_init_main` returns to `INT_Initialize`, control is transferred to `INC_Initialize` for complete, high-level initialization of the Nucleus PLUS system. `INC_Initialize` calls the `Application_Initialize` function after all other initialization is complete, and before the Nucleus PLUS scheduler is invoked.



## User Modifications

Configuring Nucleus PLUS and the PowerPC startup routines to a new target environment might require slight source modifications. Please see the Diab Data PowerPC documentation/source for a related discussion. The following files might require minor modifications.

File	Meaning
INT_FADS.S OR INT_EST.S	This is the Nucleus PLUS initialization file. Typically, this file contains most low-level initialization.
USER_MMU.C	This is the MMU initialization file. It contains the table of user-defined memory regions.

## Execution Mode

All execution in the Nucleus PLUS system is done with the MSR of the PowerPC set to 0x1032. This value enables the Machine Check state, enables the MMU, sets non-maskable interrupts to a recoverable state, and bases the vector table at address 0x0. When interrupts are enabled, the MSR is set to 0x9032 to allow External Exception interrupts. This includes initialization, task execution, and interrupt processing.

## Register Usage

Nucleus PLUS uses general purpose registers as outlined in the following table.

Register	Usage
r0	Scratch register. Not preserved across function boundaries.
r1	Stack Pointer.
r2	Reserved
r3 - r12	Temporary registers. Not preserved by functions, or across function boundaries. Holds variables whenever possible.
r3 - r10	Used for parameter and result passing.
r13	Reserved.
r14 - r31	Preserved registers. Saved when used by functions. Contains variables that cannot be placed in r3 - r12.



### Memory Usage

Nucleus PLUS, like any program, requires memory for instructions and data. Memory for instructions, global data, and static data is allocated during the link phase of building a system. During system startup, Nucleus PLUS performs a minimal amount of run-time memory allocation. All subsequent memory allocation is under the control of the application.

The linker decides where to locate memory SECTIONS in a given target environment. A memory SECTION is basically a collection of similar data elements grouped together by the linker, i.e. data, text, and bss. For example, all processor instructions are grouped together in the `".TEXT"` area. All initialized data is in the `".DATA"` area, and the uninitialized data is in the `".BSS"` area.

### Nucleus PLUS Memory Usage

Nucleus PLUS performs some minimal memory allocation after the end of the `__SP_INIT` area. This area is the starting point of the system stack and is placed just after the `".TEXT"` and `".BSS"` sections of memory. The starting addresses and sizes of each section are set up in the `DEMO.LNK` file. Nucleus PLUS allocates the following areas, which are each defined in `INT_FADS.S` or `INT_EST.S`:

- System stack area
- Timer HISR stack area

The address after the previously mentioned allocation is passed to `Application_Initialize` as the first available memory location.

### Stacks

There are principally three types of stacks in a Nucleus PLUS system, namely the system, task, and High-Level Interrupt Service Routine (HISR) stacks. The system stack is defined and allocated in `INT_FADS.S` or `INT_EST.S`. Task and HISR stacks are determined by the user during creation.

The system stack is used during initialization and while executing in the scheduling loop. Additionally, the system stack is used during execution of Low-Level Interrupt Service Routines (LISRs). The size of the system stack should take into account the worst case interrupt requirements in the system, including nested interrupt conditions.

The context of each task or HISR is saved on its respective stack. If a task or HISR is interrupted, all registers are saved on the stack. Otherwise, if a task suspends or relinquishes control as a result of a Nucleus PLUS service call, only the registers that must be preserved across function boundaries are saved. The size of task and HISR stacks must be large enough to accommodate all local variable allocation and maximum function call nesting, in addition to the space necessary to save all of the PowerPC 823 registers.



The first word of the stack of a non-executing task or HISR determines the type of stack format present. If the first word contains a value of one, an interrupt stack frame is present. Otherwise, if the first word of the stack is zero, a non-interrupt stack frame is present.

Interrupt Stack Frame:

tc_stack_pointer	->	Next available space
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved
		Stack type (Interrupt Stack Type 1)
		SRR1/MSR
		Registers r0, r3, -r8, r14-r31
		CTR
		XER
		CR
		LR
		SRR1/MSR
		SRR0/TCC_Task_Shell
		r9
		r10
		r11
		r12
		Diab/Data old SP if being saved
		Diab/Data old LR which is saved

Solicited Stack Frame:

tc_stack_pointer	->	Next available space
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved
		Stack type (Interrupt Stack Type 0)
		MSR
		Registers r14-r31
		CTR
		CR
		LR
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved

### Minimum Stack Size

Nucleus PLUS is configured to not allow task or HISR stacks that are less than 172 bytes in size. Additionally, a stack overflow condition is considered present if there are fewer than 164 bytes remaining on the non-floating point task or HISR stack.

### Tuning Task/HISR Stack Size

Internal stack checking and the service `NU_Check_Stack` check for stack overflow conditions. At the same time, the minimum amount of available stack space is monitored. This value can be used to determine how much stack space is needed by a given task or HISR.



## Interrupt Vectors

Nucleus PLUS interrupt vectors are defined in `INT_FADS.S` or `INT_EST.S` at the label `INT_Vectors`. The entire Nucleus vector table is located at address 0x0 during the linking stage. During the run-time Nucleus initialization, the IP bit of the MSR register is cleared to force all exceptions to vector from base address 0x0.

LISRs can be registered for all interrupts except system reset (0x100), and decremter (0x900). The system reset vector jumps to `INT_Initialize`. Nucleus uses the decremter for its periodic timer interrupt. The following is a list of all the MPC823FADS interrupt sources and their corresponding Nucleus vector numbers:

Interrupts	Nucleus vector number
<b>Core</b>	
Machine check	2
Alignment	6
Program error	7
System call	12
Trace	13
Software emulation	16
Instruction TLB miss	17
Data TLB miss	18
Instruction TLB error	19
Data TLB error	20
Data breakpoint	28
Instruction breakpoint	29
Peripheral breakpoint	30
Non-maskable dev port	31

Interrupts	Nucleus Vector Number
<b>SIU</b>	
IRQ 0	32
Level 0	33
IRQ 1	34
Level 1	35
IRQ 2	36
Level 2	37
IRQ 3	38
Level 3	39
IRQ 4	40
Level 4	41
IRQ 5	42
Level 5	43
IRQ 6	44
Level 6	45
IRQ 7	46
Level 7	47



Interrupts	Nucleus vector number
<b>CPM</b>	
Error	64
PC4	65
PC5	66
SMC2	67
SMC1	68
SPI	69
PC6	70
Reserved	71
Reserved	72
PC7	73
PC8	74
PC9	75
Reserved	76
Reserved	77
PC10	78
PC11	79

Interrupts	Nucleus vector number
<b>CPM</b>	
I2C	80
RISC timer table	81
Timer 2	82
Reserved	83
IDMA2	84
IDMA1	85
SDMA Channel Bus error	86
PC12	87
PC13	88
Timer 1	89
PC14	90
Reserved	91
Reserved	92
SCC2	93
USB	94
PC15	95

## Interrupt Control

Interrupts may be disabled and enabled, for brief periods, by Nucleus PLUS. The interrupts disabled in a PowerPC 823 environment are defined by `NU_DISABLE_INTERRUPTS` and `NU_ENABLE_INTERRUPTS`. By default, these are set to 0x0 and 0x8000 respectively.



### Interrupt Control Services

Nucleus PLUS allows the user to enable and disable PowerPC 823 interrupts. The values passed to the interrupt control routines represent the interrupt lockout bits to be placed in the PowerPC 823 Machine State Register (MSR). Furthermore, the interrupt lockout bits must be in the same location as they are in the MSR.

There are two default parameters to the interrupt control functions (`NU_Control_Interrupts` and `NU_Local_Control_Interrupts`), as follows:

Default Parameter	Value	Meaning
<code>NU_ENABLE_INTERRUPTS</code>	0x8000	Enables External
<code>NU_DISABLE_INTERRUPTS</code>	0x0000	Disables External

Interrupt control services may be called from application tasks, LISRs, and HISRs. Interrupt control services are not allowed from the `Application_Initialize` function. It is important to realize that interrupts enabled or disabled by `NU_Control_Interrupts` remain enabled or disabled until a subsequent call is made. There is one exception to this rule. If this function is called from an LISR in a nested interrupt condition, the previously interrupted ISR's interrupt level is restored after the current LISR finishes.

### Initial Interrupt Lockout

After initialization is complete, Nucleus PLUS places the value of `NU_ENABLE_INTERRUPTS` into the MSR. Hence, before any task or HISR executes, interrupts are enabled to this level.

### Timer Interrupt

Nucleus PLUS requires a periodic interrupt in order to provide time-oriented services such as time-slicing, service call timeouts, and application timers. Initialization and interrupt vector assignments are target specific and are typically done in the `INT_Initialize` function of `INT_FADS.S` or `INT_EST.S`. Nucleus uses the Decrementer (0x900) interrupt for its periodic timer interrupt.

### Reentrancy

Reentrant C functions are functions that do not access or rely on global or static data. Nucleus PLUS services are fully reentrant. Diab Data library functions may or may not be reentrant, depending on the function. Please review the Diab Data documentation regarding the reentrancy of the Diab Data library functions.



4

# Nucleus PLUS Control Block Offsets

Task Control Block Offsets

HISR Control Block Offsets



## Task Control Block (TCB) Offsets

The Task Control Block (TCB) contains information used to manage the execution of a task in the Nucleus PLUS system. This structure is defined in a manner that insures that most of its members are aligned on long-word boundaries.

Decimal Offset	HEX Offset	TCB Member
0	00	tc_created
12	0C	tc_id
16	10	tc_name
24	18	tc_status
28	1C	tc_delayed_suspend
32	20	tc_priority
36	24	tc_preemption
40	28	tc_scheduled
44	2C	tc_cur_time_slice
48	30	tc_stack_start
52	34	tc_stack_end
56	38	tc_stack_pointer
60	3C	tc_stack_size
64	40	tc_stack_minimum
68	44	tc_current_protect
72	48	tc_saved_stack_ptr
76	4C	tc_time_slice
80	50	tc_ready_previous
84	54	tc_ready_next
88	58	tc_priority_group
92	5C	tc_priority_head
96	60	tc_sub_priority_ptr
100	64	tc_sub_priority
104	68	tc_saved_status
108	6C	tc_signal_active
112	70	tc_entry
116	74	tc_argc
120	78	tc_argv
124	7C	tc_cleanup
128	80	tc_cleanup_info
132	84	tc_suspend_protect
136	88	tc_timer_active
140	8C	tc_timer_control
144	90	tc_signals
148	94	tc_enabled_signals
152	98	tc_signal_handler



## HISR Control Block (HCB) Offsets

The HISR Control Block (HCB) contains information used to manage the execution of a HISR in the Nucleus PLUS system. This structure is defined in a manner that insures that most of its members are aligned on long-word boundaries.

Decimal Offset	HEX Offset	HCB Member
0	00	tc_created
12	0C	tc_id
16	10	tc_name
24	18	tc_not_used_1
28	1C	tc_not_used_2
32	20	tc_priority
36	24	tc_not_used_3
40	28	tc_scheduled
44	2C	reserved
48	30	tc_stack_start
52	34	tc_stack_end
56	38	tc_stack_pointer
60	3C	tc_stack_size
64	40	tc_stack_minimum
68	44	tc_current_protect
72	48	tc_active_next
76	4C	tc_activation_count
80	50	tc_entry



# Nucleus C++

**Nucleus C++ Software**  
**Building the Nucleus C++  
Library From a DOS Prompt**  
**Building and Executing the  
C++ Demonstration System  
From a DOS Prompt**  
**System Startup**  
**Operators New and Delete**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

### Nucleus C++ Software

Nucleus C++ for the PowerPC 823 environment is shipped on CD-ROM. The CD contains all the C/C++ source code and include files for Nucleus C++, target specific assembly files, build and link command files, and an example system.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

Nucleus C++ is implemented as a C library. The Nucleus C++ library and demo rely on the following structure, and will not build correctly if the structure is modified. After installation, the directory structure will be as follows:



### Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	Diab Data V4.3B
Assembler:	Diab Data V4.3B
Linker:	Diab Data V4.3B
Debugger:	Single Step On-Chip (MPC8XX) V 7.4
Board:	Motorola MPC823FADS and EST Boards*

(\*programs for the EST are compile only)

### Building the Nucleus C++ Library From a DOS Prompt

Nucleus C++ is implemented as a library. The MS-DOS batch file PLUSPLUS.BAT contains the Diab Data assembler, compiler, linker, and archiver commands necessary to build the Nucleus C++ library. Execution of PLUSPLUS.BAT produces the Nucleus C++ library file PLUSPLUS.LIB in the \O subdirectory and rebuilds the PLUS library.

```
\ATI\PLUSPLUS > plusplus <cr>
```



## Compiler Options

The compiler options used to build the Nucleus PLUS library are used. In addition, compilation parameters that are new to Nucleus C++ are as follows:

Compilation Flag	Meaning
-Xno-exception	Tells the compiler to disable exception handling.
-Xno-implicit-templates	Tells the compiler to instantiate templates only where explicit instantiation syntax is used.

## Conditional Compilation

The Nucleus C++ library has several conditional compilation options. These conditional compilation flags enable various features within the Nucleus C++ library source. The flags are to be used in conjunction with the flags for the Nucleus PLUS kernel. They are defined using `#define` statements in the port specific header file `NUCPP.H`.

The conditional compilation options are as follows:

Compilation Flag	Meaning
<code>NU_ERROR_STRING</code>	Enables making an ASCII error string if a fatal system error occurs. This flag is applicable to the compilation of <code>ERD.C</code> , <code>ERI.C</code> , <code>ERC.C</code> , and the entire Nucleus C++ component.
<code>NU_NO_ERROR_CHECKING</code>	Disables error checking shell on creation of the timer <code>HISR</code> in <code>TMI.C</code> . It is also applicable to the Nucleus C++ library component compilation. This will result in a significant run-time performance increase and will also reduce code size.
<code>NU_DEBUG</code>	Maps the data structures used by the application and defined in <code>NUCLEUS.H</code> to the actual internal data structures in Nucleus PLUS. This option must be used for Nucleus C++.



### Building and Executing the C++ Demonstration System From a DOS Prompt

Building an application is described in the "Getting Started" chapter of the *Nucleus C++ Reference* manual. Also, specific debugger commands are documented in the target specific notes for the Nucleus PLUS kernel in previous chapters.

Assuming that `NUCPP.H` , `PLUS.LIB` and `PLUSPLUS.LIB` are accessible from their respective directory, the following commands build and link the demonstration application. The default configuration is for the MPC823 FADS board. If you are using a different MPC823 board, use the appropriate command line parameter (EST for the EST board). The default may be changed at the top of the `DEMO.BAT` batch file.

```
\ATI\PLUSPLUS\DEMO> demo <cr>
```

Typically, `*.S` and/or the `INIT.C` file of the Diab Data tools need to be modified for a particular target environment.

The file `DEMO.CPP` contains the `InitializeCppApplication` function as well as all tasks and other objects. The `DEMO.LNK` files define what needs to be linked together and where everything belongs for the corresponding board.

This batch file assembles the `INT_FADS.S` or `INT_EST.S`, compiles `DEMO.CPP` and links it all together. The `DEMO.LNK` file contains information necessary for the linker. This file should be used as a template for any user-created linker command file.

A file named `DEMO.ELF` is produced by the batch file. Make sure all files are compiled with full debug information (`-g` option with the Diab compiler)

The demo may be loaded and executed in the same manner as the Nucleus PLUS demo.

Set a watch on the `"screen"` variable. It will appear as a text array in your *Watch* window. Each time you stop demo execution, the `"screen"` array will be updated and will appear similar to the screen output of the PLUS port if you had a serial driver. The array will display the status of the variables you normally watch with the PLUS port.

Please refer to Chapter 5 of the *Nucleus C++ Reference* manual to familiarize yourself with the demo application. This will explain good places to set breakpoints and serves as an excellent tutorial for the Nucleus C++ operating system.



## System Startup

Nucleus C++ utilizes the Diab Data startup functions. Note: the application is not allowed to have a function named `main`.

`INT_Initialize` is responsible for setting up control registers, the interrupt vector table, several system stack areas, and typically the periodic timer interrupt. After this low-level initialization is complete, control is transferred to `INC_Initialize` for complete, high-level initialization of the Nucleus PLUS system.

`INC_Initialize` calls the `Application_Initialize` function for initialization of the Nucleus C++ components. The `InitializeCppApplication` function is called by `Application_Initialize` after all other initialization is complete, and before the Nucleus PLUS scheduler is invoked.

Note that before `InitializeCppApplication` is called, the Diab Data specific `__init` routine is called. This iterates the static constructor list. This routine is called after the Nucleus C++ system is initialized, supporting static RTOS objects. This support is new since the release of the *Nucleus C++ Reference* manual.

## Operators New and Delete

The following Memory Management options are selected for the demonstration system running on the Software Development Systems Debugger. These options are selected in `NUCPP.H`.

Parameter	Meaning
<code>NU_HEAP_START_VALID</code>	Set to 1. <code>NU_HEAP_START</code> is a valid address.
<code>NU_HEAP_START</code>	Points to the beginning of the heap. Set to <code>first_available_memory</code> , which is initialized during startup.
<code>NU_HEAP_SIZE</code>	Specifies the number of bytes in the heap as 65536 or 64K Bytes of free store.
<code>NU_HEAP_REENTRANT</code>	Specifies whether <code>malloc</code> is reentrant. Not applicable for this port.
<code>NU_HEAP_SUSPEND</code>	Specifies how threads suspend during memory allocation if necessary. Possible values are <code>NU_NO_SUSPEND</code> , <code>NU_PRIORITY</code> , and <code>NU_FIFO</code> . Set to <code>NU_PRIORITY</code> by default.
<code>NU_HEAP_NUMBER_POOLS</code>	Specifies the number of " <code>NU_HEAP_SIZE</code> " byte memory pools created. Not applicable for this port.
<code>NU_HEAP_ZERO_MEMORY</code>	Specifies whether the new operator zeroes out the allocated memory before it is returned to the user. Set to <code>TRUE</code> by default.

Please refer to the *Nucleus C++ Memory Management* document for information on custom settings.





# 6

## Nucleus NET

Introduction

Building the Nucleus NET  
Library From a DOS Prompt

Defines Used for Compilations

Nucleus NET Interface  
Routines

Overview of The  
Demonstration System

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

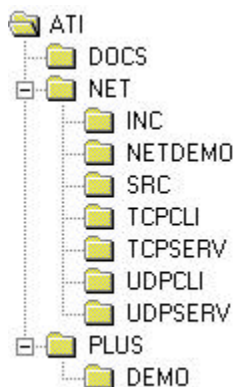


### Introduction

This chapter discusses the installation of Nucleus NET for Power PC 823 using Diab Data tools. Nucleus PLUS must be installed and working properly, before Nucleus NET is installed. It is recommended that you build and execute the Nucleus PLUS demonstration application prior to building Nucleus NET. The building of the Nucleus PLUS library, `PLUS.LIB` is discussed in Chapter 2.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

To install the Nucleus NET software, execute `SETUP.EXE` from the distribution software. After installation the directory structure will be as follows:



Nucleus NET depends on this directory structure, and will not build correctly unless the batch files and/or source files are modified. If you purchased a driver (Ethernet, SLIP, PPP, etc.) from Accelerated Technology, you should refer to that chapter of the manual for specific instructions. If no driver was purchased you can build the Nucleus NET library and Nucleus NET demo applications, but you will be unable to execute the demo applications.

There are several files required from the Nucleus PLUS installation. Verify that `NUCLEUS.H` is located in the corresponding `\PLUS` directory, and `PLUS.LIB` is located in the `\PLUS\O` directory.



## Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	Diab Data V4.2B
Assembler:	Diab Data V4.2B
Linker:	Diab Data V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V 7.4
Board:	Motorola MPC823FADS and EST Boards*

(\*programs for the EST board are compile only)

## Building The Nucleus NET Library From a DOS Prompt

Once the Nucleus NET files have been installed, and the driver (if any) has been installed, you will be ready to build the Nucleus NET library, `NET.LIB`.

The MS-DOS batch file contains the Diab Data assembler, compiler, and librarian commands necessary to build the Nucleus NET library. To build the NET library, simply enter the following command at an DOS prompt.

```
\ATI\NET> net <cr>
```

Execution of the file `NET.BAT` will produce the `O\NET.LIB` library file. It is assumed that interrupts will be used.

## Compilation Switches Used

There are several compilation switches that are used to create each object file. The following lists the compilation switches used for the Diab Data compiler.

Compiler Switch	Meaning
-c	Specifies a compile to <code>.OBJ</code> without linking.
-v	Specifies that source level debugging is on.
-g	Specifies to add debugging information.
-tPPC860ES	Creates code for PowerPC ELF format.



## Defines Used for Compilations

Nucleus NET library has several conditional compilation options. Conditional compilation requests are specified with the `-D` compilation option. The following conditional compilation flags enable various features within the Nucleus NET library.

Compilation Flag	Description
INTERRUPT	Enables compilation of Nucleus NET for interrupt mode. Use of Nucleus NET in polling mode is not recommended, as performance will be very poor. Also, not all drivers provided by Accelerated Technology support polling mode.
PLUS	Enables compilation of Nucleus PLUS calls only. If PLUS is not defined the Nucleus RTX service calls will be used. (NOTE: Nucleus RTX is no longer supported with Nucleus NET.)
PACKET	If PACKET is defined, packets are transmitted as soon as they are ready, otherwise they are placed in a queue when they are ready. Use of Nucleus NET in polling mode is not recommended, as performance will be very poor. Also, not all drivers provided by Accelerated Technology support polling mode.

**Note 1)** The `INTERRUPT` flag will affect only some of the network files as these files are the only ones whose behavior varies with the mode of operation. The files that are affected are:

ARP.C	EQUEUE.C	NET.C
TCP.C	TCPVARS.C	TOOLS.C
USER.C	Any Driver	

**Note 2)** The `PACKET` flag will only affect `NET.C` and the driver files.

**Note 3)** The MPC823 driver only supports the `NON-PACKET` mode of transmission.



## TARGET.H

TARGET.H is target specific header file that contains configuration information for Nucleus NET. The file is well commented. Please read the comments within TARGET.H for more information about the options it contains.

## Nucleus NET Interface Routines

For a discussion of the Nucleus NET interface routines, see the *Nucleus NET Reference Manual*.

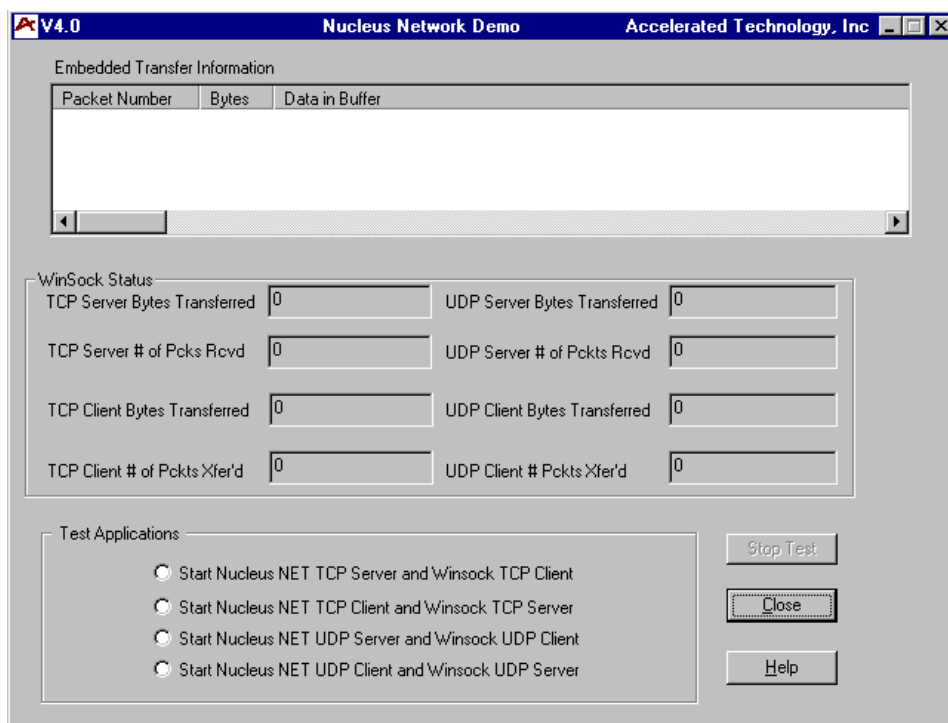
## Overview of the Demonstration System

Nucleus NET contains four demos. The demo shows either the TCP or UDP protocol working as a server or a client. The demos are designed to communicate with the Winsock application \NET\NETDEMO\NETDEMO.EXE which runs on any Windows 95/98 or Windows NT machine connected to the network. NETDEMO.EXE can be run in 4 different modes so that it can communicate with any of the embedded demos. The two applications will communicate over the echo port which is port 7. The server echos back data sent to it by the client. The demos shipped with NET are compile only. They require a driver that can be written by the user or purchased from Accelerated Technology Inc. If a driver was purchased from ATI, build and run those demos instead of these NET demos. However, the functionality of the demos are the same. Please read this section and the chapter regarding the driver purchased.

## Using the Nucleus NetDemo Application

The main screen for the Nucleus NetDemo Application is shown in Figure 6.1. There are three fields that are of interest. The fields are *Embedded Transfer Information*, *Winsock Status*, and *Test Applications*.





**Figure 6.1 The NetDemo Main Screen**

The *Embedded Transfer Information* field holds information that was received by the Winsock `Recv` command issued from the Nucleus NET application. The information is divided into three parts, *Packet Number*, *Number of Bytes in Packet* and the *Data Buffer*. The Scroll Bar may be used to view all of the information. The test application will accept 2000 packets, and will continue to accept more packets. It will, however, start to remove the first packet from sight, and continue to add the latest packet at the bottom of the scroll status.

The *Winsock Status* shows the status of the Winsock application that is running during each test. The information will display “*total bytes sent/received*”, and “*total # number packets sent/received*”, depending on which TEST is being performed.

The bottom section of Figure 6.1 shows the *Test Applications*, which consists of *Start TCP Server*, *Start TCP Client*, *Start UDP Server*, and *Start UDP Client*. To perform each of these test applications, refer to the embedded Nucleus NET applications that were provided with your Nucleus NET code.

### Settings Common to all the NET Demos

All of the NET demos include code to set up the driver. The code is different for each driver but each driver will have the same code in all four demos. Here is a listing of driver initialization code from a compile only NET demo:

```
devices[0].dv_name = "DRIVER_0";
devices[0].dv_hw.ether.dv_irq = DEMO_IRQ;
devices[0].dv_hw.ether.dv_io_addr= DEMO_IO_ADDR;
devices[0].dv_hw.ether.dv_shared_addr = 0;
devices[0].dv_init = Init;
devices[0].dv_flags = 0;
memcpy (devices[0].dv_ip_addr, cli_ip_addr, 4);
memcpy (devices[0].dv_subnet_mask, subnet, 4);
```

Each demo also has a `printf` symbol that is defined. This function is for debugging purposes. The default is a function that is stubbed out and gives no debugging info. `printf` can be set to the name of a user supplied debugging function.

### TCP Server Demo

TCPSEV.C is the TCP Server Demo. It accepts a limited number of connections from clients and echos the data the client sends. Before building the demo modify the following lines of code:

```
#define DEMO_IRQ          5
#define DEMO_IO_ADDR      0x0300L
#define DEMO_MAX_CONNECTIONS 10
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
```

DEMO\_IRQ and DEMO\_IO\_ADDR are driver specific defines that are not used in this compile only demo. DEMO\_MAX\_CONNECTIONS is the number of pending connections the server will accept. DEMO\_SERVER\_IP\_ADDR is the IP address of the board.

Use this command line to build the TCP Server demo. Remember, each driver contains its own demos so the path the batch file that builds each demo may be slightly different.

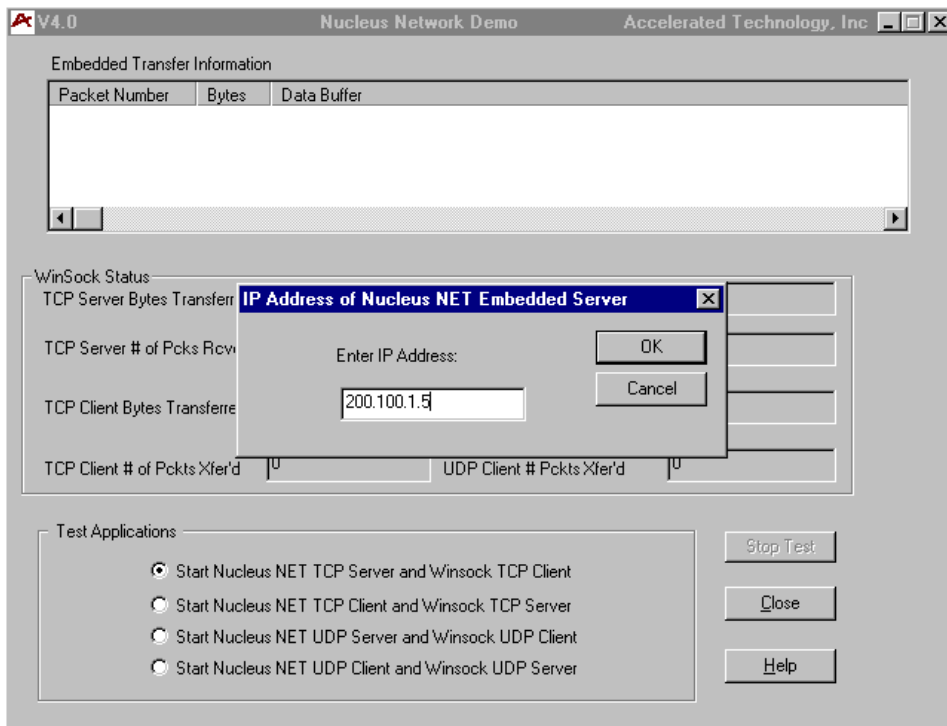
```
\ati\NET\TCPSEV> TCPSEV <CR>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then \ATI\NET\TCPSEV\O\TCPSEV.ELF is downloaded to the board just like the PLUS demo. For more details about downloading code see Chapter 2. Always start the server application first. This means start the embedded application before NETDEMO.EXE.



## Nucleus Target Specific Notes PowerPC 823 Diab Data Tools

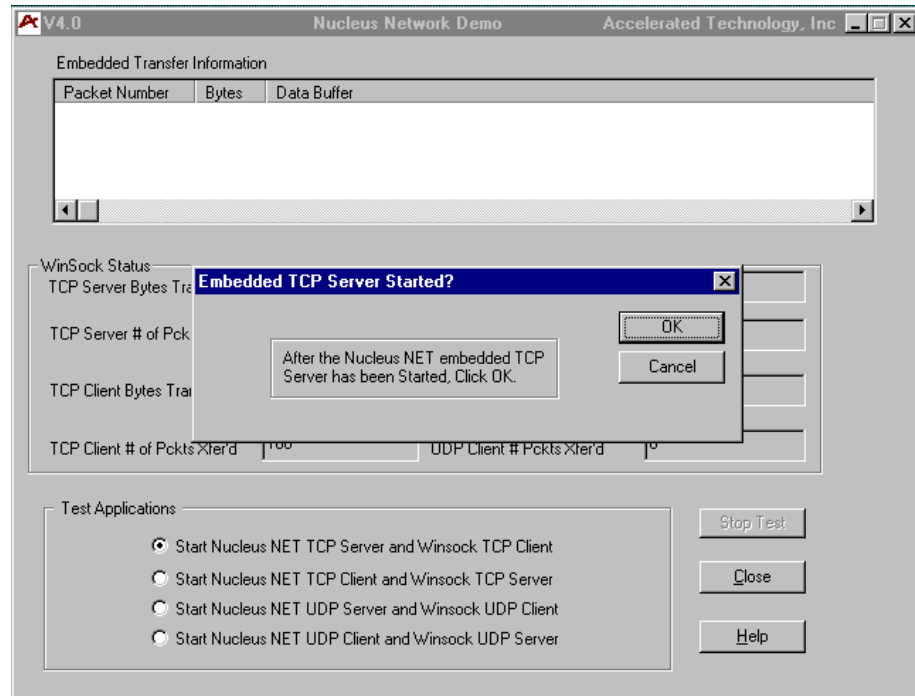
Once NETDEMO.EXE is started, click on the 'Start TCP Server Application'. The Pop-up window shown in Figure 6.2 below will appear. This will allow you to type in the IP address of the Nucleus NET TCP Server. After inputting your specific IP address, click OK.



**Figure 6.2 Configuring the Nucleus NET TCP Server**



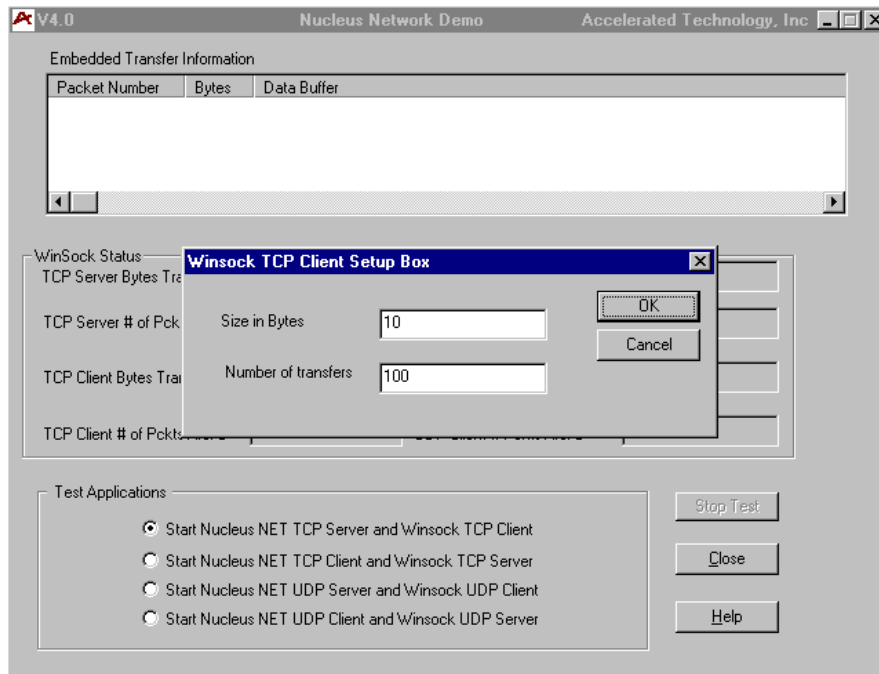
The next window will be displayed as shown in Figure 6.3. This Pop-up window will display a message to start the Nucleus NET embedded TCP Server. Once the embedded TCP Server is started click *OK*.



**Figure 6.3 Starting the Nucleus NET TCP Server**

## Nucleus Target Specific Notes PowerPC 823 Diab Data Tools

You should configure the Winsock TCP Client as shown in Figure 6.4. This Pop-up window allows you to enter the *Size in Bytes* of the TCP client packet, and the *Number of transfers* that the TCP Client packet will be transferring. Click *OK* , after you have entered the values.



**Figure 6.4 The Size in Bytes/Number of Transfers Screen**



Data should be transferring at this time and you should see status being updated. The screen should appear similar to the screen displayed in Figure 6.5.

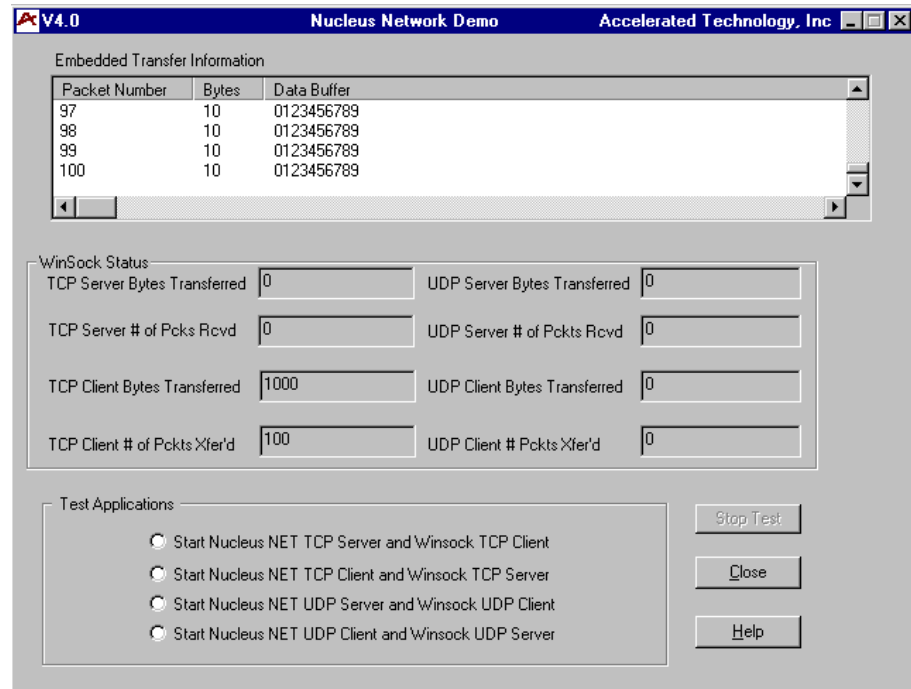


Figure 6.5 TCP Server/Client Status Update Screen

## TCP Client Demo

TCPCLI.C is the TCP Client Demo. It connects to a server and sends *"This is TCP test packet # X"* where X is the number of packets that have been sent. Before building the demo modify the following lines of code:

```
#define printf          PRINTF
#undef  USE_HOST_TABLE
#define DEMO_IRQ        5
#define DEMO_IO_ADDR    0x0300L;
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_NAME "server_name"
```



If `USE_HOST_TABLE` is defined the demo will look for the servers IP using a call to `NU_Get_Host_By_Name`, otherwise the IP address must be supplied by the user. If `USE_HOST_TABLE` is defined there must be a machine to help resolve the DNS or the name must be in `HOST.C` before the `NET` library is compiled. `DEMO_IRQ` and `DEMO_IO_ADDR` driver specific defines that are not used in this compile only demo. `DEMO_SERVER_IP_ADDR` needs to be set to the IP address of the computer running `NETDEMO.EXE` and the name of that computer should be assigned to `DEMO_SERVER_NAME`. The IP address of the board should be assigned to `DEMO_CLIENT_IP_ADDR`.

Use this command line to build the TCP Client Demo. Remember, each driver contains its own demos so the path and the batch file that builds each demo may be slightly different.

```
\ATI\NET\TCPCLI> TCPCLI <CR>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then `NET\TCPCLI\O\TCPCLI.ELF` is downloaded to the board just like the `PLUS` demo. For more details about downloading code see Chapter 2. Always start the server application first. This means execute `NETDEMO.EXE` before starting the embedded demo.



Once NETDEMO.EXE is started, click on the 'Start TCP Client' button. This will display a Pop-up window as shown in Figure 6.6. Click *OK*. Then start the Nucleus NET embedded TCP client. Once *OK* is clicked, the Winsock TCP server is started. This insures that TCP client data is not being transmitted before the Winsock TCP server is prepared to accept data.

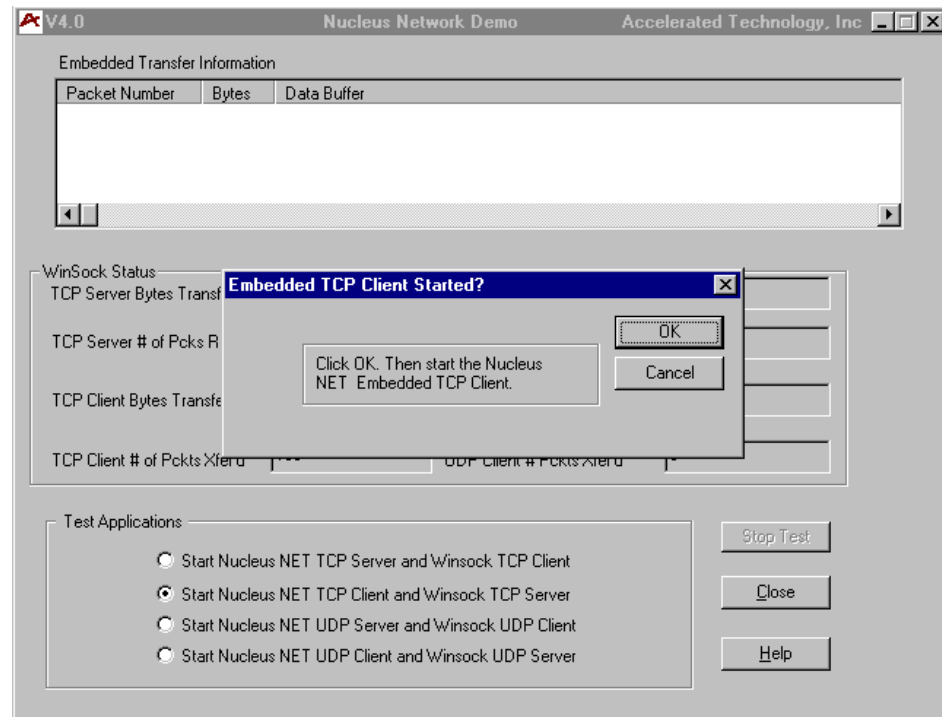
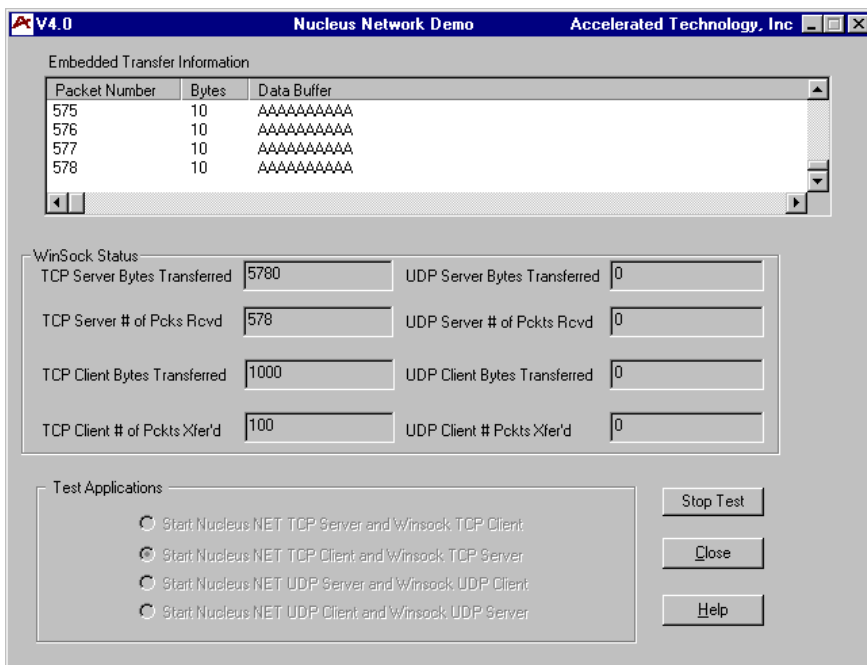


Figure 6.6 Starting the TCP Client

Figure 6.7 is an example of how the screen should appear while the test is running.



**Figure 6.7 Testing the TCP Client Transfer Rate**

## UDP Client Demo

UDPCLI.C is the UDP Client Demo. It connects to a server and sends packets containing strings of capital As. Before building the demo modify the following lines of code:

```
#define DEMO_IRQ          5
#define DEMO_IO_ADDR      0x0300L
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_NAME  "server_name"
```

DEMO\_IRQ and DEMO\_IO\_ADDR are driver specific defines that are not used in this compile only demo. The settings are very similar to those used in TCPCLI.C. DEMO\_SERVER\_IP\_ADDR needs to be set to the IP address of the computer running NETDEMO.EXE and the name of that computer should be assigned to DEMO\_SERVER\_NAME. The IP address of the board should be assigned to DEMO\_CLIENT\_IP\_ADDR.



Use this command line to build the UDP Client Demo. Remember, each driver contains its own demos so the path the the batch file that builds each demo may be slightly different.

```
\ATI\NET\UDPCLI> udpcli <cr>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then \ATI\NET\UDPCLI\O\UDPCLI.ELF is downloaded to the board just like the PLUS demo. For more details about downloading code see Chapter 2. Always start the server application first. This means execute NETDEMO.EXE before starting the embedded demo.

Once NETDEMO.EXE is started, click on the 'Start UDP Client' button. This will display a Pop-up window as shown in Figure 6.8. Click *OK*. Then start the Nucleus NET embedded UDP client. Once OK is clicked, the Winsock UDP server is started. This insures that UDP client data is not being transmitted before the Winsock UDP server is prepared to accept data.

An example of how the screen should appear while the test is running, is shown below in Figure 6.9.

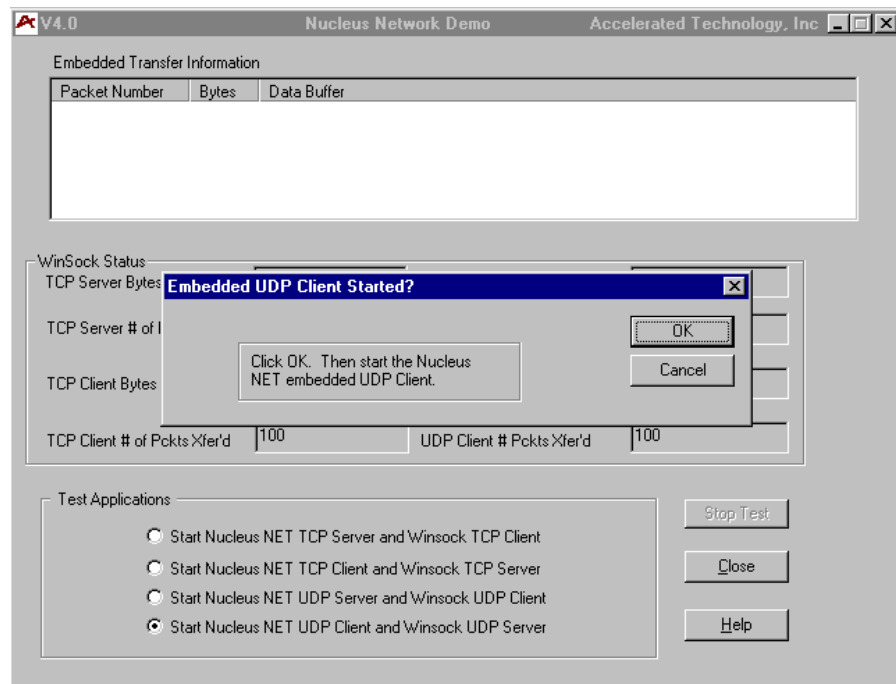
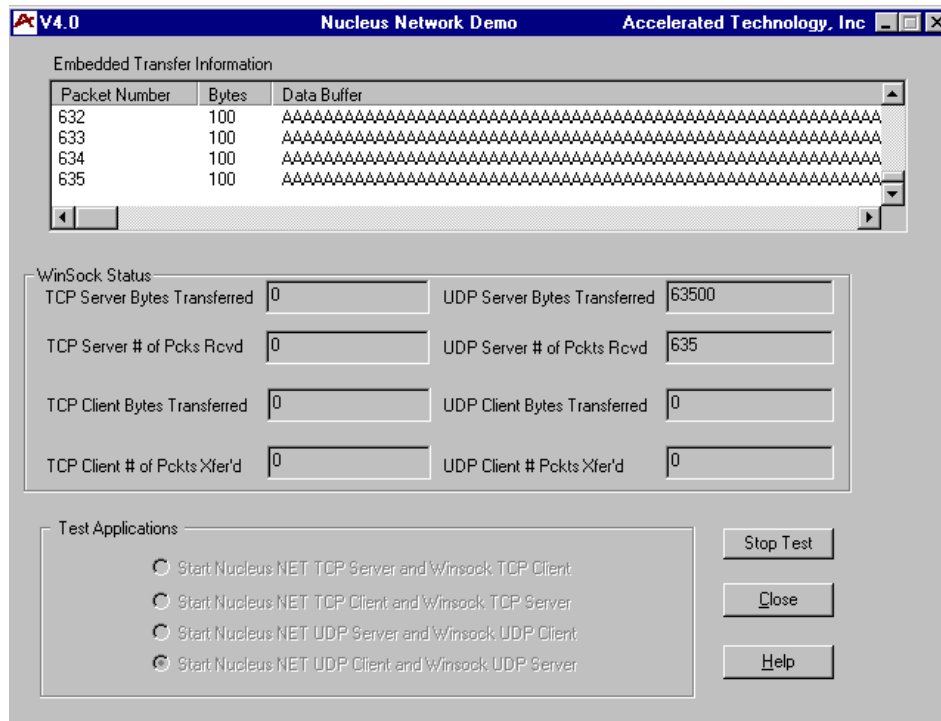


Figure 6.8 Starting the UDP Client



### Figure 6.9 Testing UDP Client Transfer Rate

## UDP Server Demo

UDPSERV.C is the UDP Server Demo. It accepts connections from clients and echos the data the client sends. Before building the demo, modify the following lines:

```
#define DEMO_IRQ 5
#define DEMO_IO_ADDR 0x0300L
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_CLIENT_NAME "client_name"
#define DEMO_SERVER_NAME "server name"
```

DEMO\_IRQ and DEMO\_IO\_ADDR are driver specific defines that are not used in this compile only demo. DEMO\_CLIENT\_ADDR needs to be set to the IP address of the PC running NETDEMO.EXE. DEMO\_CLIENT\_NAME is the name of this computer running NETDEMO.EXE. DEMO\_SERVER\_IP\_ADDR needs to be set to the IP address of the embedded board. DEMO\_SERVER\_NAME is the name of the board.



Use this command line to build the UDP Server Demo. Remember, each driver contains its own demos so the path and the batch file that builds each demo may be slightly different.

```
\Ati\NET\UDPSERV> udpserv <cr>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then \NET\UDPSERV\O\UDPSERV.ELF is downloaded to the board just like the PLUS demo. For more details about downloading code see Chapter 2. Always start the server application first. This means start the embedded application before NETDEMO.EXE.

Once NETDEMO.EXE is started, click on 'Start UDP Server Application'. The pop-up window shown in Figure 6.10 below will appear. This will allow you to type in the IP address of the Nucleus NET UDP Server. After inputting your specific IP address, click OK.

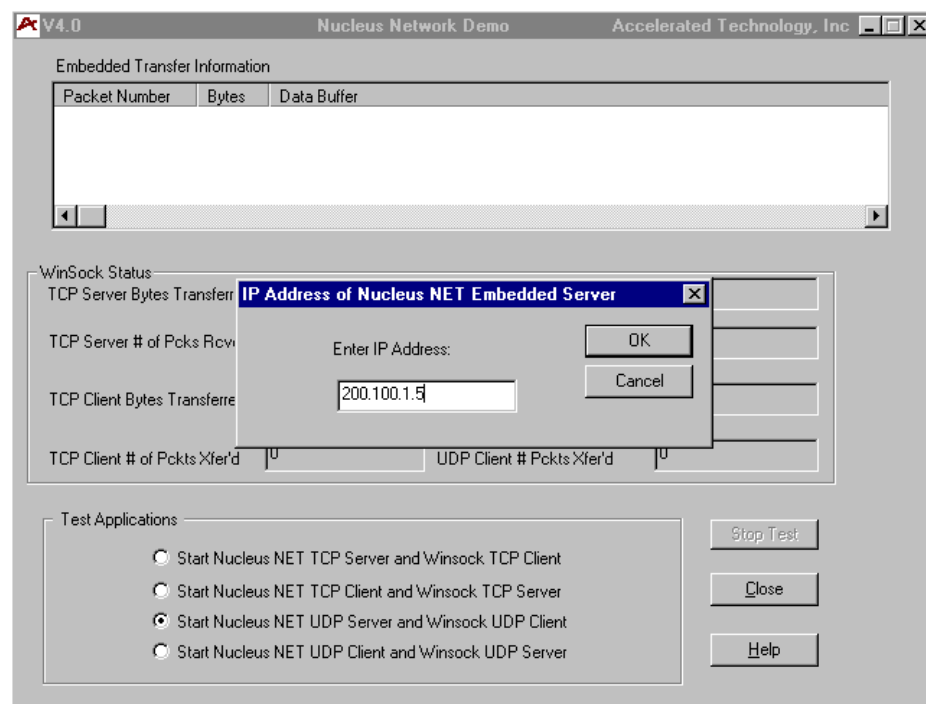
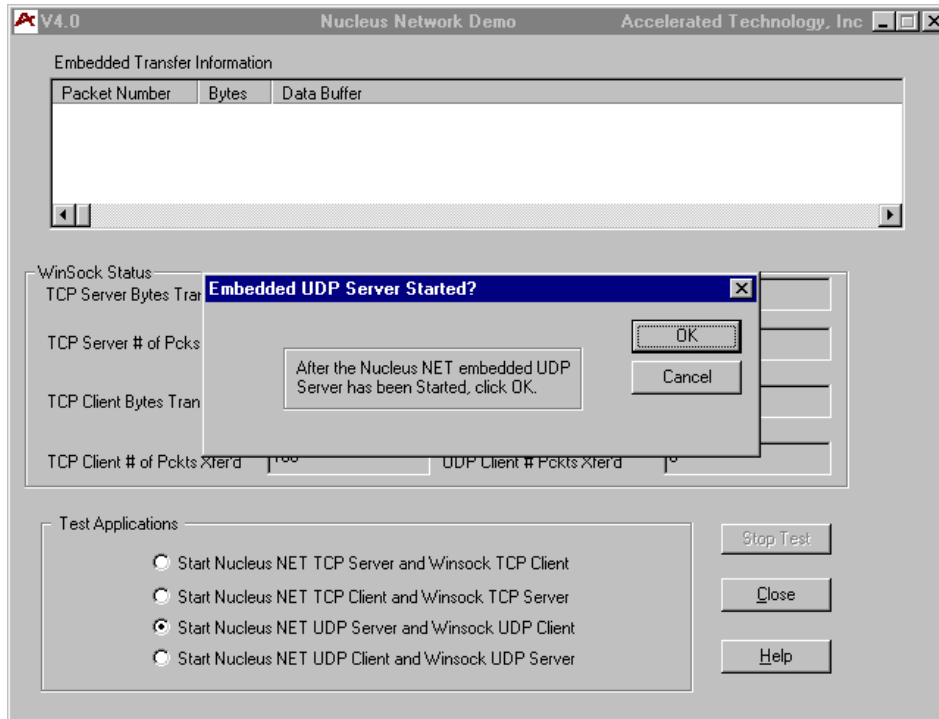


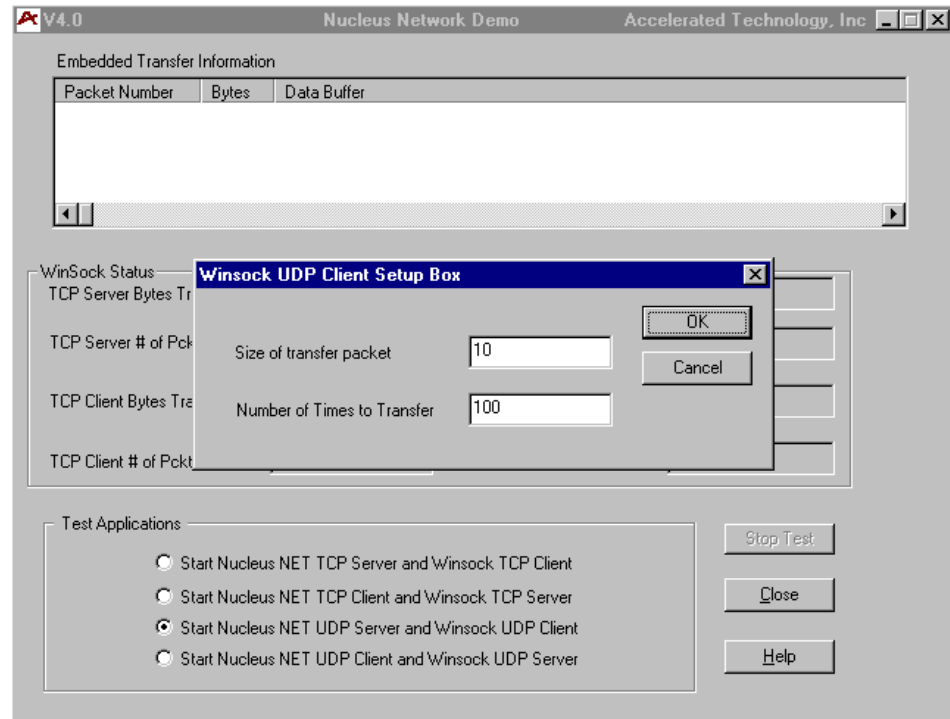
Figure 6.10 Starting the UDP Server Application

The next window, shown in Figure 6.11, will be displayed. This Pop-up window will displays a message to start the Nucleus NET embedded UDP Server. Once the embedded UDP Server is started, click *OK*.



**Figure 6.11 The UDP Server Start message**

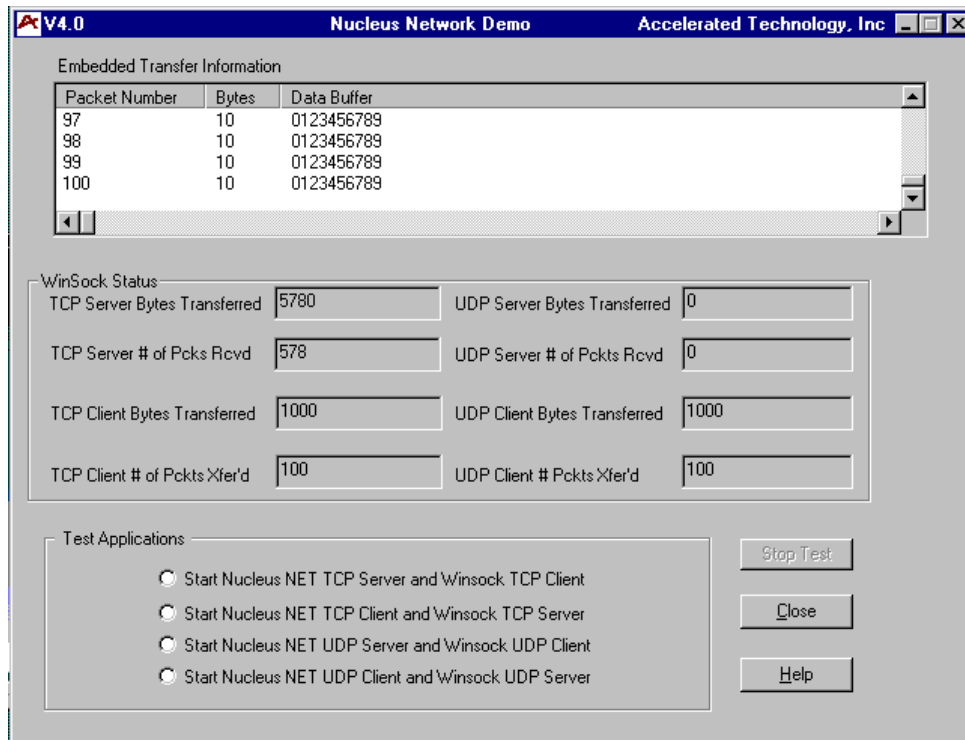
The next Pop-up window is displayed, which will allow you to configure the Winsock UDP Client and is shown in Figure 6.12 below. This Pop-up window will allow you to enter the *Size in Bytes* of the UDP client packet, and the *Number of Times to Transfer* by the UDP Client packet. Click *OK* after you have entered the appropriate values.



**Figure 6.12 Size of Packets/Number of Times to Transfer Screen**

## Nucleus Target Specific Notes PowerPC 823 Diab Data Tools

Data should be transferring at this time, and you should see status being updated. The screen should look similar to Figure 6.13.



**Figure 6.13 UDP Server/Client Transfer Status Information**



## Stopping the Test Application

To stop the test application while running, click the *Stop Test* button as shown in Figure 6.14.

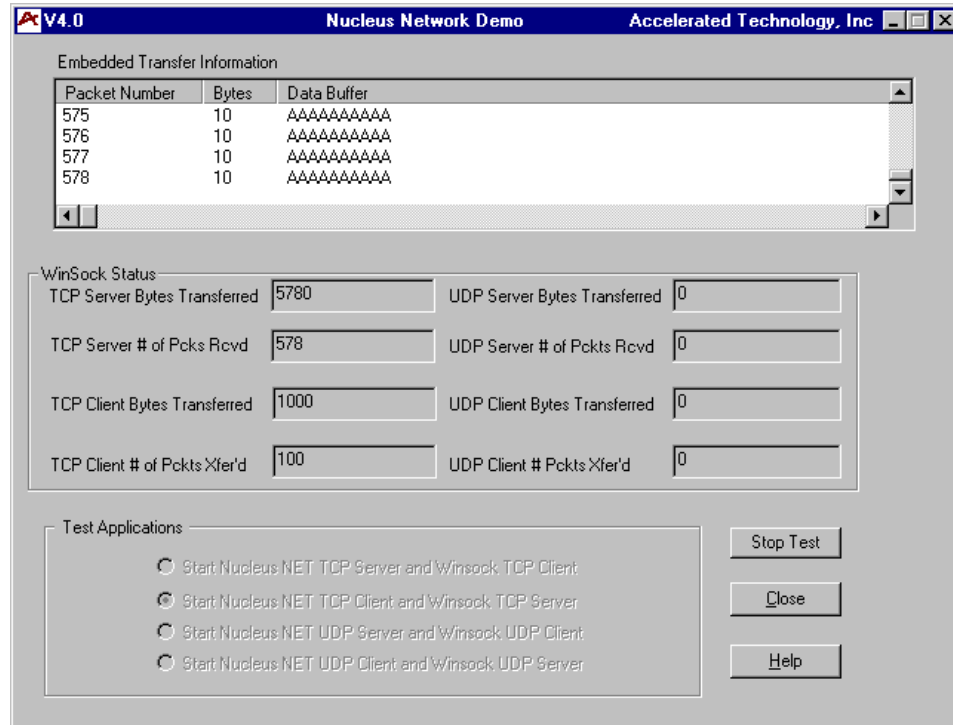


Figure 6.14 Stopping the Test

## Exiting the Program

To exit the program, click the *Close* button.





# 7

## Nucleus Ethernet Driver

### Introduction

**Building the Nucleus Ethernet Library  
from a DOS Prompt**

**Building and Executing the Ethernet  
Demonstration System**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



### Introduction

This chapter briefly describes the installation of the MPC823 Ethernet Controller driver for Nucleus NET. This driver is designed to work with two boards - the Motorola FADS 823, with DIAB compiler, and SDS BDM debugger and the Embedded Support Tools (EST) SBC8XX with DIAB compiler, and SDS BDM debugger or EST BDM debugger.

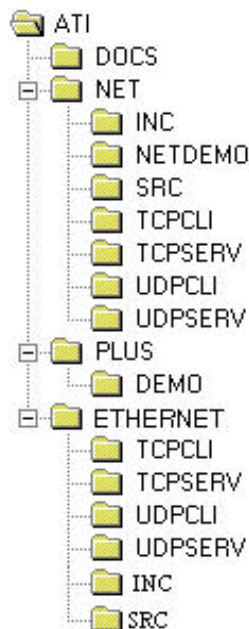
PQUICC stands for Motorola Power Quad Integrated Communications Controller. The MPC823 does not have a Quad Integrated Communication Controller. The name PQUICC is kept to make code porting easier. The Serial Communication Controller 2 is configured as an ethernet controller.

Nucleus PLUS and Nucleus NET must be installed, before the Nucleus MPC823 ethernet driver is installed. It is recommended that you build the Nucleus PLUS and the Nucleus NET libraries before attempting to build the Nucleus MPC823 Ethernet applications. Installation of Nucleus NET is covered in the Nucleus NET chapter.

Execute the `SETUP.EXE` program to install the driver and associated files to your hard drive.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

After installation the directory structure will be as the follows:



## Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	Diab Data V4.2B
Assembler:	Diab Data V4.2B
Linker:	Diab Data V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V 7.4
Board:	Motorola MPC823FADS and EST Boards*

(\*programs for the EST boards are compile only)

## Building the Nucleus Ethernet Library From a DOS Prompt

Once the Nucleus products have been installed, you will be ready to build the Nucleus MPC823 Ethernet library, `ETHERNET.LIB`.

One board is supported, the Motorola FADS 823.

The MS-DOS batch file contains the Diab Data assembler, compiler, and librarian commands necessary to build the Nucleus MPC823 Ethernet library. The default compiles for the MPC823 FADS board. To build the Ethernet library, simply enter the following command at an DOS prompt.

```
\ATI\ETHERNET> ethernet <cr>
```

Execution of the file `ETHERNET.BAT` will produce the `O\ETHERNET.LIB` library file. It is assumed that interrupts will be used.

## Description of Driver Files

The file `PQUICC.C` contains the interface routines to Nucleus NET, a utility routine for initialization, an interrupt service routine, a FIFO routine for transmitting packets, and a routine for receiving packets.

Each device on a network has a unique address called a hardware address. If two devices share a hardware address both devices will confuse each other. MPC823 sets the hardware address within `PQUICC.C`. To ensure your address is unique from anyone else who is running `PQUICC`, change any one of the following lines in `PQUICC.C` to be any arbitrary values:

```
device->dev_mac_addr[0] = 0x66;
device->dev_mac_addr[1] = 0x55;
device->dev_mac_addr[2] = 0x44;
device->dev_mac_addr[3] = 0x33;
device->dev_mac_addr[4] = 0x14;
device->dev_mac_addr[5] = 0x11;
```



The file `PQUICC.H` contains the necessary symbolic constants and data structure type definitions. The most important data type is the PDA structure. A pointer of this type, called `MPC823`, is used to overlay the memory area that maps to the registers of the Communication Processor Module (CPM). Once this pointer is initialized to point at the address of the internal RAM area, all operations on the MPC823 registers are performed by accessing the appropriate field in this data structure.

## Building and Executing the Ethernet Demonstration System

The MPC823 Ethernet demos operate like the NET demos except the MPC823 demos use the Ethernet driver. See Chapter 6 for information about the NET demos. This section contains information about what was changed to make the NET demos use the MPC823 Ethernet driver.

**NOTE:** Each demo must be modified for your network. Please edit each demo source file and make modifications for your network prior to building the demos.

Each demo is built using the commands listed here:

```
\ATI\ETHERNET\TCPCLI> tcpcli <cr>
\ATI\ETHERNET\TCPSERV> tcpserv <cr>
\ATI\ETHERNET\UDPCLI> tcpcli <cr>
\ATI\ETHERNET\UDPSERV> tcpserv <cr>
```

Each demo may be loaded and executed like the PLUS demo. Each of the demos include a prototype for the Ethernet driver initialization function. It is listed below.

```
extern STATUS MPC823_Init(DV_DEVICE_ENTRY *device);
```



Each driver contains driver is initialized code like the following:

```

devices[0].dv_name = "MPC823_0";
devices[0].dv_hw.ether.dv_irq = DEMO_IRQ;
devices[0].dv_hw.ether.dv_io_addr = DEMO_IO_ADDR;
devices[0].dv_hw.ether.dv_shared_addr = 0;
devices[0].dv_init = MPC823_Init;
devices[0].dv_flags = 0;
memcpy (devices[0].dv_ip_addr, serv_ip_addr, 4);
memcpy (devices[0].dv_subnet_mask, subnet, 4);
if (NU_Init_Devices(devices, 1) != NU_SUCCESS)
{
    printf("NU_Init_Devices failed in UDP_Server_Task.\n");
    DEMO_Exit(6);
}

```

`dv_name` is a unique name for that device. `dv_init` is set to the MPC823 initialization function. `cd_flags` is set to 0. The **memcpy** commands copy the demo's IP address and subnet mask into the device structure. `dv_hw.ether.dv_io_addr`, `dv_hw.ether.dv_irq`, and `dv_hw.ether.dv_shared_addr` are all unused by PQUICC. They are left to set be used for example code if another driver might be used. The last statement initializes the device driver.





# Nucleus GRAFIX

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

## Introduction

**Building the Nucleus GRAFIX GS  
Library From a DOS Prompt**

**Building and Executing the  
GRAFIX GS Demonstration  
System From a DOS Prompt**

**Building the Nucleus GRAFIX WT  
Library From a DOS Prompt**

**Building and Executing the  
GRAFIX WT Demonstration  
System From a DOS Prompt**

**Display Modes**

**GS Configuration**

**WT Configuration**



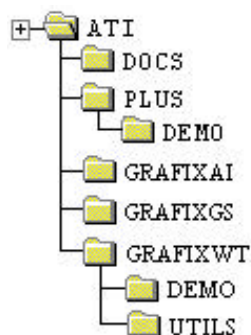
### Introduction

The following chapter will discuss the installation of Nucleus GRAFIX, and building the Nucleus GRAFIX library.

**NOTE:** Nucleus PLUS must be installed, before Nucleus GRAFIX is installed. It is recommended that you build and execute the Nucleus PLUS demonstration application before attempting to build Nucleus GRAFIX.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

To install the Nucleus GRAFIX software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



GRAFIX depends on this directory structure and will not build correctly unless the batch files are modified.

### Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	Diab Data V4.3B
Assembler:	Diab Data V4.3B
Linker:	Diab Data V4.3B
Debugger:	Single Step On-Chip (MPC8XX) V 7.4
Board:	N/A – compile only



## Building the Nucleus GRAFIX GS Library From a DOS Prompt

Build the GRAFIX Services library, `grafixGS.LIB`, by executing the following command:

```
\ATI\GRAFIXGS> grafixgs <cr>
```

## Building and Executing the GRAFIX GS Demonstration System From a DOS Prompt

Build the GRAFIX Services demo, `GSDEMO.ELF`, by executing the following command:

```
\ATI\GRAFIXGS\DEMO> gsdemo <cr>
```

Download this demo in the same manner as you did the Nucleus PLUS demo. In a few minutes you should see several different test screens cycling on the display. If you do not see this displayed, attempt to reload the program.

## Building the Nucleus GRAFIX WT Library From a DOS Prompt

Build the GRAFIX WT library, `grafixWT.LIB`, by executing the following command:

```
\ATI\GRAFIXWT> grafixwt <cr>
```

## Building and Executing the GRAFIX WT Demonstration System From a DOS Prompt

Build the GRAFIX WT demo, `WTDemo.ELF`, by executing the following command:

```
\ATI\GRAFIXWT\DEMO> wtdemo <cr>
```

Download this demo in the same manner as you did the Nucleus PLUS demo. After the program is loaded. Execute the program. In a few minutes you should see a Windows like display with the typical Nucleus demo program output on the LCD. This consists of a Windows like background with several lines of text scrolling up the screen. If you do not see this displayed, attempt to reload the program.



### Display Modes

This release has four display modes available to the user:

Name	Description
cLCD (also cLCD8)	640x480 8-bit color
cLCD1	640x480 1-bit monochrome
cLCD2	640x480 2-bit color
cLCD4	640x480 4-bit color

cLCD is what has been tested since that is the only panel we have. Note that these are color modes and not grey scale. Furthermore, the display is programmed to be an active display with a single scan, 12 bit interface (NEC p/n NL6448AC33-10) - there are some significant differences in how the LCD controller is programmed for other panels. The user should refer to chapter 18 of the Power PC 823 User's Manual.

### GS Configuration

The file CONFIG.H can be found in the GRAFIXGS directory. The file contains information pertaining to the setup of the Nucleus Grafix Rendering Services. The file contains the following defines:

Define	Description
MNT	Undefined for the MPC823 version.
CYCLE	Used with in the GSDEMO. If defined the demo will cycle continuously. If not defined it will require input events(Pointing device/keyboard) to move through frames of the demonstration. By default for the MPC823 this is defined.
_USE_NU_ALLOC	Uses NUCLEUS Memory allocation routines. By default this is defined.
MAX_DEFWIN_X	Sets the Screen X win MAX size. By default this is not used.
MAX_DEFWIN_Y	Sets the Screen Y win Max Size. By default this is not used.
BPP	This define is used for Bits Per Pixel. It is set up for 8 bit, 16 bit and 24 bit. By default this define is not used.



## WT Configuration

The file WTCONFIG.H can be found in the GRAFIXWT directory. The file contains multiple defines pertaining to the setup of the Nucleus Grafix Windowing Toolkit. The file contains the following defines:

Define	Description
_MAC_OS	Set the Look and Feel of the Windowing Toolkit to Macintosh Look and Feel. Undefined by default.
_WIN95_OS	Set the Look and Feel of the Windowing Toolkit to Windows 95 Look and Feel. This is the default look and feel.
_USE_BACK_BUFFERS	Use Back Buffers within Windowing Toolkit. By default this is defined.
McDEBUG	A tool for debugging event handling between dialog windows. It is undefined by default.
DEF_SCROLL_ATTACH	Define Makes all LIST, TEXT and EDIT Boxes get created with a scroll bar attached. If undefined the VSCROLLBAR attribute will need to be set with each control by using the SetCtrlAttribute function. It is defined by default.
_MONOCHROME	Define for monochrome and grayscale settings. This define is used to determine the screen and window settings. By default this item is not defined.
DEF_DISPLAY_MODE	Used to determine if display mode is color or monochrome. 1 is for COLOR and 0 is for monochrome.
DEF_DESK_PATTERN	Define for the Desktop pattern number. By default this is 26 for COLOR and 29 for MONOChrome. 26 and 29 are the Default fill patterns used in the Graphic Services library.
DEF_DESK_PATTERN_FORE_COLOR	Desktop Foreground color. Default is White for Color and monochrome.
DEF_DESK_PATTERN_BACK_COLOR	Desktop Background Color. Default is Red for Color and Black for Monochrome.
DEF_SHOW_BACKGROUND	Set the Desktop to Transparent mode if it is set to a one. it is zero by default.
DEF_DESK_TEXT_FORE_COLOR	Desktop Text fore ground color. Default is Black.
DEF_DESK_TEXT_BACK_COLOR	Desktop Text background color. Default is White
DEF_DESK_PICTURE	Desktop window's back drop. This would contain the name of the image file to load for the desktop window's backdrop.



Define	Description
DEF_DESK_TILE_PICTURE	Desktop Tile for the backdrop of the desktop window. 1 is if tiling is done to the backdrop and 0 for if the backdrop for the desktop window is not tiled.
DEF_BUTTON_STYLE	Defines the button style. These are the possible button setting: 0 = FLAT, 1= THIN, 2=THICK, 3=WIN95. Default of the button style is set to 0 for flat.
DEF_WINDOW_MIN_WIDTH	Defines the minimum width of the window. Default of the windows minimum width is set to 100.
DEF_WINDOW_MIN_HEIGHT	Defines the minimum height of the window. Default of the windows minumum height is set to 120.
DEF_WINDOW_BUTTON_WIDTH	Defines the default width dimension for the buttons on the title bar. Default width of the buttons on the title bar is 13.
DEF_WINDOW_BUTTON_HEIGHT	Defines the default height dimension for the buttons on the title bar. Default height of the buttons on the title bar is 13.
DEF_WINDOW_TITLE_BAR_HEIGHT	Defines the default height of the window's title bar. Default of the window's title bar height is 20.
DEF_WINDOW_SUB_TITLE_BAR_HEIGHT	Defines the default height of the window's sub title bar. Default of the window's sub title bar height is 20.
DEF_WINDOW_STATUSBAR_HEIGHT	Defines the default height of the window's status bar. Default of the window's status bar height is 20.
DEF_WINDOW_MENUBAR_HEIGHT	Defines the default height of the window's menu bar. Default of the window's menu bar height is 20.
DEF_SYSTEM_MENUBAR_HEIGHT	Defines the default height of the system's menu bar. Default of the system's menu bar height is 20.
DEF_WINDOW_BORDER_WIDTH	Defines the default border width of a window. Default of the window's border width is 4.
DEF_WINDOW_X_SPACING	Defines the default text spacing in the X direction for a window. Default of the X spacing for text in the window is 20.
DEF_WINDOW_Y_SPACING	Defines the default text spacing in the Y direction for a window. Default of the Y spacing for text in the window is 20.
DEF_SCROLL_BUTTON_WIDTH	Defines the default scroll button width dimensions. These dimensions apply to both windows and controls that use scroll bars. Default scroll button width is 14.
DEF_SCROLL_BUTTON_HEIGHT	Defines the default scroll button height dimensions. These dimensions apply to both windows and controls that use scroll bars. Default scroll button height is 14.



Define	Description
DEF_WND_FORE_COLOR	Defines the default fore ground color for all window's but the desktop window. Default foreground color for the window's that are not the desktop is Black.
DEF_WND_BACK_COLOR	Defines the default back ground color for all window's but the desktop window. Default background color for the window's that are not the desktop is White.
DEF_CLIENT_FORE_COLOR	Defines the default foreground color for window's client area. Default foreground color for the window's client area is Black.
DEF_CLIENT_BACK_COLOR	Defines the default background color for window's client area. Default background color for the window's client area is White.
DEF_TITLE_FORE_COLOR	Defines the default foreground color for window's title bars. Default foreground color for the window's title bar is Black.
DEF_TITLE_BACK_COLOR	Defines the default background color for window's title bars. Default background color for the window's title bar is White.
DEF_CONTROL_FORE_COLOR	Defines the default foreground color for all control's that need to be filled. Default foreground color for all control's that need to be filled is Black.
DEF_CONTROL_BACK_COLOR	Defines the default background color for all control's that need to be filled. Default background color for all control's that need to be filled is White.
DEF_BUTTON_FORE_COLOR	Defines the default foreground color for push buttons. Default foreground color for the push buttons is Black.
DEF_BUTTON_BACK_COLOR	Defines the default background color for push buttons. Default background color for the push buttons is White.
DEF_HILITE_FOREGROUND	Defines the default foreground color for highlighting text in menus, listbox, combobox, and edit box controls. Default foreground color for highlighting text is White.
DEF_HILITE_BACKGROUND	Defines the default background color for highlighting text in menus, listbox, combobox, and edit box controls. Default background color for highlighting text is Black.
DEF_3D_SHADOW_COLOR	Defines the default shadow color for drawing a 3d rectangle around an object. If you make 3d shadow color and ltsource color Black then it turns off the 3d effect. Default setting for the 3d shadow color is Black.
DEF_3D_LTSOURCE_COLOR	Defines the default light source color for drawing a 3d rectangle around an object. If you make 3d shadow color and ltsource color Black then it turns off the 3d effect. Default setting for the 3d ltsource color is White.
DEF_DISABLE_FORE	Defines the default foreground color for text in a disabled control. Default setting for text foreground in a disabled control is Black.



Define	Description
DEF_DISABLE_BACK	Defines the default background color for text in a disabled control. Default setting for text background in a disabled control is White.
MOUSE_TRACK_SCALE	Defines the default tracking speed of the mouse in use. Default setting of the tracking speed of the mouse in use is 1.
MOUSE_FORE_COLOR	Defines the default foreground color of the mouse cursor. Default setting of the foreground color of the mouse cursor is Black.
MOUSE_BACK_COLOR	Defines the default background color of the mouse cursor. Default setting of the background color of the mouse cursor is White.
DEF_CURSOR_BLINK_RATE	Defines the default text cursor blinking rate. Default setting for the text cursor blinking is 9.
DEF_DELAY_TIMER	Defines the delay between single click to auto repeat mouse down events. Default setting for the delay between single click to auto repeat mouse down events is 15.
DEF_CLICK_TIMER	Defines the threshold for double mouse clicks. Default setting for the threshold for double mouse clicks is 10.
DEF_DRAWING_FORE_COLOR	Defines the foreground system drawing color. Default setting for the foreground system drawing color is Black.
DEF_DRAWING_BACK_COLOR	Defines the background system drawing color. Default setting for the background system drawing color is White.
DEF_STICKY_MENUS	Defines the default menu settings. Default setting is 0.
DEF_AUTO_HELP	Defines the default help settings. Default setting is 0.
DEF_CONTROL_FILL_COLOR	Defines the default control fill colors. Default setting for the control fill color is Black.
DEF_CONTROL_FILL_PATTERN	Defines the default control fill pattern. Default setting for the control pattern is 0, this is for no pattern.



# Nucleus TFTP Server

## Introduction

**Building the TFTP Server Library  
From a DOS Prompt**

**Building and Executing the TFTP  
Server Demonstration System From  
a DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

### Introduction

This chapter will discuss the installation of the Nucleus TFTP Server source code, building the Nucleus TFTP Server library, and building the demonstration application.

**NOTE:** Nucleus PLUS, NET, and a driver must be installed before Nucleus TFTP Server is built. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus TFTP Server. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see the chapter '*Nucleus Ethernet Driver*' of this manual.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

To install the Nucleus TFTP Server software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 823 port using the Diab Data Tools must be installed in a common directory (C:\ATI by default). TFTP Server will require PLUS and NET to be present in this directory structure. If ETHERNET is not present, then the demo will need to be modified to use the substitute driver. TFTP Server expects to be within this directory structure and will not build otherwise. After installation, the directory structure will be as follows:



### Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step 7.4
Board:	Motorola MPC823 FADS and EST Boards*

(\*programs for the EST board are compile-only.)

## Building the Nucleus TFTP Server Library From a DOS Prompt

To build the TFTP library, the user should execute the following command:

```
\ATI\TFTPSERV> tftpserv <cr>
```

This will compile all the files of the TFTP Server library and put all of them into the library file, `TFTPSERV.LIB`

## Building and Executing the TFTP Server Demonstration System From a DOS Prompt

`TFTPSDEM.C` is the Nucleus TFTP Server demo application. This server waits for data from a client and echos the data back to the client. The demo will need to be modified to use another driver. If you have purchased another driver from Accelerated Technology, see the corresponding chapter for information about how to modify the demo to use that driver. Regardless of what driver is used, modify the following lines of code before building the demo:

```
#define printf                PRINTF
#define DEMO_IRQ              5
#define DEMO_IO_ADDR          0x0300L;
#define DEMO_SERVER_IP_ADDR   {100,200,200,200}
```

The symbol `printf` is defined to a function that has been stubbed out. This can be redefined as a function to help debug this demo. `DEMO_IRQ` and `DEMO_IO_ADDR` are not used by the `ETHERNET` driver. They have been left to help users who use other drivers that may need that information. `DEMO_SERVER_IP_ADDR` is the IP address of the board. To compile and link the TFTP demonstration program, execute the following command:

```
\ATI\TFTPSERV\DEMO> tftpsdem <cr>
```

Download the code to the board using the same process described for the `PLUS` port. For more information see Chapter 2. Once the program has been started on the board, use a TFTP Client application to upload a small file to the evaluation board. If the program calls the function `DEMO_EXIT` with the value 0 then the program executed correctly.





# Nucleus TFTP Client

## Introduction

**Building the Nucleus TFTP Client Library From a DOS Prompt**

**Building and Executing the TFTP Client Demonstration System From a DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

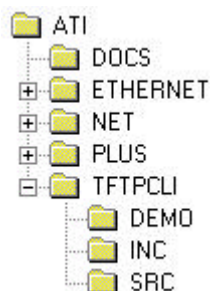
### Introduction

The following chapter will discuss the installation of the Nucleus TFTP Client, and building the Nucleus TFTP library.

**NOTE:** Nucleus PLUS, NET, and a driver must be installed, before Nucleus TFTP is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus TFTP.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure will be as follows:



### Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step 7.4
Board:	Motorola MPC823 FADS and EST Boards*

(\*programs for the EST board are compile-only.)

## Building the Nucleus TFTP Client Library From a DOS Prompt

The batch file `TFTPCLI.BAT` is provided for building the TFTP library and the batch file `TFTPDEMO.BAT` builds the demo program.

To build the TFTP library, execute the following command:

```
\ATI\TFTPCLI> tftpcli <cr>
```

## Building and Executing the TFTP Client Demonstration System From a DOS Prompt

The TFTP demo application must be configured for use on your system. There are two IP addresses that must be changed. In `TFTPDEMO.C`, search for the following two lines:

```
# define DEMO_SERVER_IP_ADDR = {100, 200, 300, 400};
# define DEMO_CLIENT_IP_ADDR = {100, 200, 300, 400};
```

The first IP address is the address of a TFTP server on your network. The second is the IP address of the TFTP client application. Modify these addresses for use on your network.

The `TFTPDEMO.BAT` builds the TFTP source files and TFTP demo application and links it all together to produce `TFTPDEMO.ELF`, using the following command prompt:

```
\ATI\TFTPCLI\DEMO> tftpdemo <cr>
```

The `TFTPDEMO.ELF` file should be downloaded to a evaluation board using SingleStep BDM debugger as described for the PLUS port. See Chapter 2 for more information.

When executed the demo first tries to establish a connection with the TFTP server. Once the connection is established the client attempts to put a file named `TEST.TXT` to the server.

If the file is successfully sent to the server, the TFTP demo will then attempt to get it back. The TFTP demo then performs a comparison of the file sent with the one retrieved to make sure that they are identical.





# Nucleus SNMP

## Introduction

### Rebuilding Other Nucleus Libraries

### Building the Nucleus SNMP Library from a DOS Prompt

### Building and Executing the SNMP Demonstration System From a DOS Prompt

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

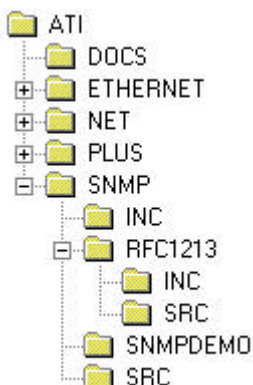
### Introduction

This chapter will discuss the installation of the Nucleus SNMP source code, building the Nucleus SNMP library, and building the demonstration application.

**NOTE:** Nucleus PLUS, NET, and a driver must be installed before Nucleus SNMP is built. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus SNMP. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see the chapters '*Nucleus Ethernet Driver*' of this manual.

**NOTE:** Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 12 for more information.

All Nucleus products associated with this Motorola PowerPC 823 port using the Diab Data Tools must be installed in a common directory (C:\ATI by default). SNMP will require PLUS and NET to be present in this directory structure. If ETHERNET is not present, then the demo will need to be modified to use the substitute driver. SNMP expects to be within this directory structure and will not build otherwise. After installation, the directory structure will be as follows:



## Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step 7.4
Board:	Motorola MPC823 FADS and EST Boards*

(\*programs for the EST board are compile-only.)

## Rebuilding other Nucleus Libraries

SNMP manages the statistics that are logged from other parts of the network stack. Nucleus NET and the driver must be built differently so these libraries will report this information to Nucleus SNMP. `NET_S.LIB` and `ETHERNET_S.LIB` are the NET and ETHERNET libraries built to log SNMP statistics. Each is kept in the directory with the original library. A library built to log SNMP statistics must be linked with the SNMP library. Hence, when verifying the driver demos or a Nucleus NET product that does not use SNMP they will need to be linked with the usual libraries (`NET.LIB` and `ETHERNET.LIB`).

However, the SNMP demo application looks for the SNMP variations of the libraries it needs. `NET_S.BAT` and `ETHERNET_S.BAT` will build the corresponding SNMP enabled libraries. These can be run by hand if needed. It may be easier to run `SNMP.BAT`. `SNMP.BAT` will build `NET_S.LIB` and `ETHERNET_S.LIB` if needed.

## Building the Nucleus SNMP Library From a DOS Prompt

Before building the SNMP library, you must edit the file `SNMP_CFG.C`. `SNMP_CFG.C` is located in `\ATI\SNMP\SRC`. There is an array of IP addresses that control what addresses the SNMP manager will respond to. Do not change the size of this array. Replace the placeholder entries with the IP address of the machines you want to use to query the embedded board. For example, change "Host1" to a label that is meaningful for your environment. Keep the field "public". Change the hex number, `0xC0C86432UL`, to the hex equivalent of the IP address for that machine.

The batch file `SNMP.BAT` will build the Nucleus SNMP library.

```
\ATI\SNMP> snmp <cr>
```

This will create `\SNMP\O\SNMP.LIB`.



### Building and Executing the SNMP Demonstration System from a DOS Prompt

The demonstration application is simply an TCP Server demonstration that has an SNMP task running in the background. Hence, the SNMP demo (SNMPAPP.C) code works the same way the NET TCP Server demo (TCPSERV.C) works. SNMPAPP.C accepts a limited number of connections from clients and echos the data the client sends. Before building the demo modify the following lines of code:

```
#define DEMO_IRQ                5
#define DEMO_IO_ADDR            0x0300L
#define DEMO_MAX_CONNECTIONS    10
#define DEMO_SERVER_IP_ADDR     {100,200,300,400}
```

DEMO\_IRQ and DEMO\_IO\_ADDR are driver specific defines that are not used in this compile only demo. DEMO\_MAX\_CONNECTIONS is the number of pending connections the server will accept. DEMO\_SERVER\_IP\_ADDR is the IP address of the board.

The batch file SNMPAPP.BAT is provided to build the SNMP demo application. To build the demo application execute the following command:

```
ATI\SNMP\SNMPDEMO> snmpapp <cr>
```

The result will be an ELF format executable, \ATI\SNMP\SNMPDEMO\O\SNMPAPP.ELF. SNMPAPP.ELF can be loaded with the SDS SingleStep BDM 8xx debugger. Start the debugger and select SNMPAPP.ELF. Start the SNMPAPP.ELF on the embedded board. Once the application is executing you can try to ping it from a machine on the network. If the response to the ping is received then you can connect from your SNMP manager.



# Nucleus EDE

Configuration for Nucleus  
EDE

Demonstration System

Exclude From Build

Starting a Debugger Through  
EDE

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



### Configuration for Nucleus EDE

The PLUS EDE project was configured to enable the user to compile for the PowerPC 823 FADS board or the PowerPC 823 EST board.

The library configuration can be set for the PowerPC 823 FADS board or the PowerPC 823 EST board. Make sure the project is selected as the active project. This is done by right-clicking on the project and selecting '*Set as Active Project*'.

Click on the '*Build*' menu, select '*Set Active Configuration*'.

Choose one of the following configurations:

Plus - Win32 FADS Board

Plus - Win32 EST Board

Click *Ok*.

**NOTE:** Some projects may only have one configuration, which is all that is needed for that product.

### Demonstration System

The Nucleus products demos can be built in multiple configurations. Check the configurations for each demo you are attempting to build. The configurations should be self-explanatory.

Doing the following can change the project configurations:

Make sure the desired project is selected as the active project. This is done by right-clicking on the project and selecting '*Set as Active Project*'.

Click on the '*Build*' menu, select '*Set Active Configuration*'.

Choose one of the configurations.

Click *Ok*.

A simpler way of jumping between configurations is by adding the configuration window to your toolbar. This is done as follows:

Click on the '*Tools*' menu, select '*Customize...*'

Click the '*Toolbars*' tab

Click on the '*Build*' checkbox, so that the '*Build*' toolbar is selected.

Click *Ok*.



You may also want to select dependencies. As an example, the `PlusPlus_Demo` depends on the `PLUS` library and `PLUSPLUS` library first being built.

Therefore, you may set the dependencies by doing the following:

Make sure the '*PlusPlus\_Demo files*' project is selected as the active project.

Click on the '*Project*' menu, select '*Dependencies...*'

Click on the *PLUS* and *PLUSPLUS* checkboxes so that *PLUS* and *PLUSPLUS* are selected.

Click *Ok*.

### Setting the Tool Directory

In order to set your `<TOOLS DIRECTORY>` macro, click on the *EDE Project Settings* toolbar shortcut. Click on the *Directories* tab. In the text window to the right of the *Tools Directory*, insure the path to your Diab Data directory is correct.

This path must include the drive, "diab", and then finally the diab version, for example: `"c:\diab\4.3b"`.

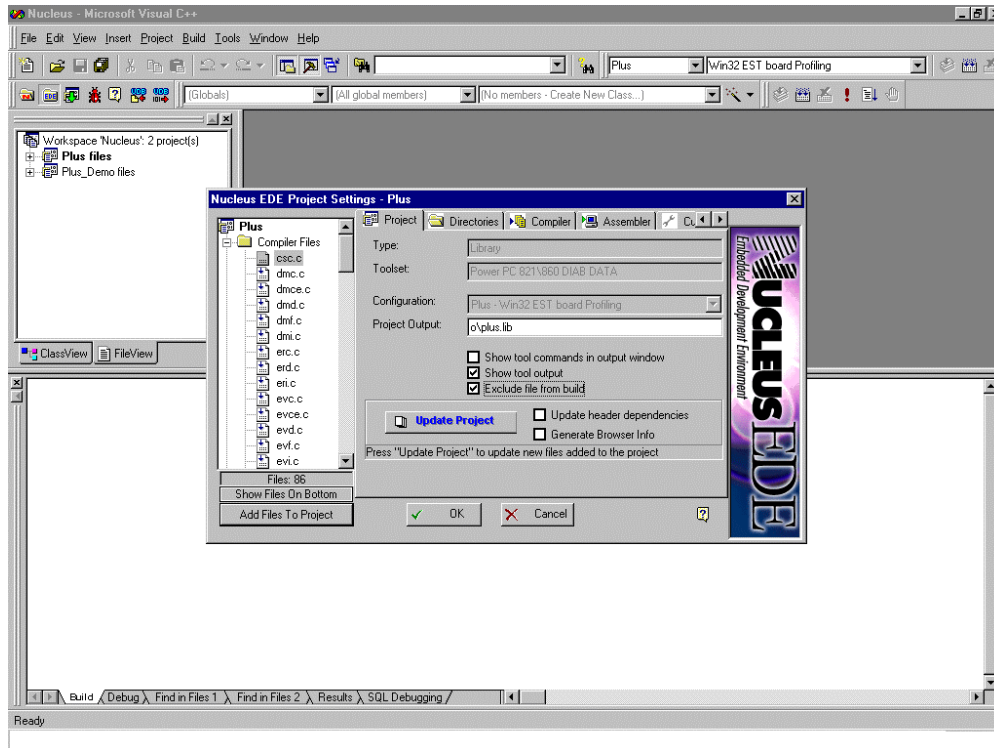


### Exclude From Build

In some cases, a file may need to be excluded from the build. To accommodate this situation, Nucleus EDE provides a *"Exclude from build"* feature. This allows certain files of the project to be ignored during the build.

To set custom settings for a file:

Click on the EDE *"Project Settings"* toolbar shortcut. Click on the appropriate file in the *File/View* window. Click on the *"Project"* tab. Check the *"Exclude from build"* box.



Exclude from Build Screen



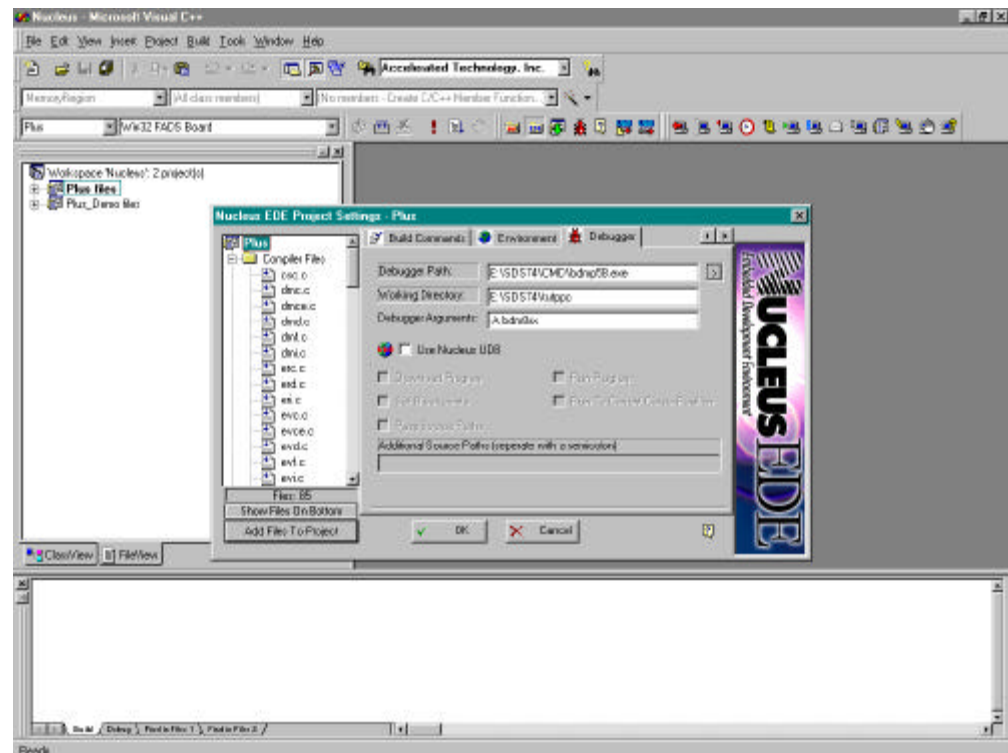
## Starting a Debugger through EDE

Nucleus EDE provides a shortcut button that will automatically start your debugger. In the case that you change debuggers, these setting will need to be manually changed.

To change the debugger properties:

Click on the Nucleus EDE toolbar shortcut *"Project Settings."* Then, click on the *"Debugger"* tab. Enter the appropriate values for the *"Debugger Path," "Working Directory,"* and the *"Debugger Arguments."*

For example, to use the SDS Debugger:



## Starting the SDS Debugger

